

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE ED

OBJECT MODULE PLACED IN ED.OBJ

COMPILER INVOKED BY: :FO: ED.PLM DATE(MARCH 16, 1982) XREF OPTIMIZE(3) DEBUG

```

1      $ TITLE('CP/M-86 2.0 --- ED')
      ED:
      DO:

      /* MODIFIED FOR .PRL OPERATION MAY, 1979 */
      /* MODIFIED FOR OPERATION WITH CP/M 2.0 AUGUST 1979 */
      /* modified for MP/M 2.0 June 1981 */
      /* modified for CP/M 1.1 Oct 1981 */

2 1    declare
      mpmproduct literally '01h', /* requires mp/m */
      cpm3        literally '30h'; /* requires 3.0 cp/m */

3 1    declare plm label public; /* entry point for plm86 interface */

      /* THE FOLLOWING COMMANDS CREATE ED.COM AND ED.CMD:

      wa $1.plm
      attach b 5
      b:se eof $1.plm
      vax $1.plm $$san\batch smpacmd $1 date(22 Oct 81)\
      b:isl4
      ERA $1.MOD
      era $1
      era $1.obj
      :f1:PLM80 $1.PLM debug PAGewidth(132) $3
      :f1:link $101.obj,$1.obj,:f1:plm80.lib to $1.mod
      :f1:locate $1.mod code(0100H) stacksize(100) map print($1.trn)
      :f1:cpm
      b:objcpm $1
      attach b 1

```

CP/M-86 v.1.1 (Vol. II)
 COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # C81-162

the following VAX commands were used to create ED.CMD

```

$ asm86 scd1.a86 debug xref
! scd1 does a jump to the plm code
$ plm86 'p1'.plm 'p2' 'p3' 'p4' optimize(3) debug
$ link86 scd1.obj,'p1'.obj to 'p1'.lnk
$ loc86 'p1'.lnk od(sm(data,code,data,stack,const)) od(sm(code(0))) ss(stack(+16))
$ h86 'p1'

```

followed by the gencmd command

```
gencmd ed data[bidB,m80,xffff]
```

where 1db is the start of the constant area / 16 from ED.MP2

*/

```

/* DECLARE                                8080 Interface
JMP EDCommand - 3 (TO ADDRESS LXI SP)
EDJMP BYTE DATA(OC3H),
EDADR ADDRESS DATA(.EDCommand-3); */

```

```

/*****
*
*      B D O S   I N T E R F A C E
*
*****/

```

```

4  1      mon1:
        procedure (func,info) external;
5  2          declare func byte;
6  2          declare info address;
7  2      end mon1;

8  1      mon2:
        procedure (func,info) byte external;
9  2          declare func byte;
10 2          declare info address;
11 2      end mon2;

12 1      mon3:
        procedure (func,info) address external;
13 2          declare func byte;
14 2          declare info address;
15 2      end mon3;

16 1      declare cmdrv    byte    external; /* command drive */
17 1      declare fcb (1)  byte    external; /* 1st default fcb */
18 1      declare fcb16 (1) byte    external; /* 2nd default fcb */
19 1      declare pass0    address external; /* 1st password ptr */
20 1      declare len0     byte    external; /* 1st passwd length */
21 1      declare pass1    address external; /* 2nd password ptr */
22 1      declare len1     byte    external; /* 2nd passwd length */
23 1      declare tbuff (1) byte    external; /* default dma buffer */

24 1      DECLARE
        BDISK    BYTE    EXTERNAL, /* BOOT DISK 0004H */
        MAXB     ADDRESS EXTERNAL, /* MAX BASE 0006H */
        BUFF (128) BYTE    EXTERNAL, /* BUFFER 0080H */
        SECTSHF  LITERALLY '7', /* SHL(1,SECTSHF) = SECTSIZE */
        SECTSIZE LITERALLY '80H'; /* SECTOR SIZE */

25 1      BOOT: PROCEDURE ;
26 2          call mon1(0,0); /* changed for HP/M-86 version */
        /* SYSTEM REBOOT */
27 2      END BOOT;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

/* ED : THE CP/M CONTEXT EDITOR */

/* COPYRIGHT (C) 1976, 1977, 1978, 1979, 1980, 1981
DIGITAL RESEARCH
BOX 579 PACIFIC GROVE
CALIFORNIA 93950

Revised:

07 April 81 by Thomas Rolander
21 July 81 by Doug Huskey
29 Oct 81 by Doug Huskey
10 Nov 81 by Doug Huskey

*/

28 1 DECLARE COPYRIGHT(*) BYTE DATA
(' COPYRIGHT (C) 1981, DIGITAL RESEARCH ');

29 1 declare date(*) byte data ('12/81');

/* COMMAND	FUNCTION
A	APPEND LINES OF TEXT TO BUFFER
B	MOVE TO BEGINNING OR END OF TEXT
C	SKIP CHARACTERS
D	DELETE CHARACTERS
E	END OF EDIT
F	FIND STRING IN CURRENT BUFFER
H	MOVE TO TOP OF FILE (HEAD)
I	INSERT CHARACTERS FROM KEYBOARD UP TO NEXT <ENDFILE>
J	JUXTAPOSITION OPERATION - SEARCH FOR FIRST STRING, INSERT SECOND STRING, DELETE UNTIL THIRD STRING
K	DELETE LINES
L	SKIP LINES
M	MACRO DEFINITION (SEE COMMENT BELOW)
N	FIND NEXT OCCURRENCE OF STRING WITH AUTO SCAN THROUGH FILE
O	RE-EDIT OLD FILE
P	PAGE AND DISPLAY (MOVES UP OR DOWN 24 LINES AND DISPLAYS 24 LINES)
Q	QUIT EDIT WITHOUT UPDATING THE FILE
R<FILENAME>	READ FROM FILE <FILENAME> UNTIL <ENDFILE> AND INSERT INTO TEXT
S	SEARCH FOR FIRST STRING, REPLACE BY SECOND STRING
T	TYPE LINES
U	TRANSLATE TO UPPER CASE (-U CHANGES TO NO TRANSLATE)
W	WRITE LINES OF TEXT TO FILE
X<FILENAME>	TRANSFER (XFER) LINES TO FILE <FILENAME>
Z	SLEEP FOR 1/2 SECOND (USED IN MACROS TO STOP DISPLAY)
<CR>	MOVE UP OR DOWN AND PRINT ONE LINE

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

IN GENERAL, THE EDITOR ACCEPTS SINGLE LETTER COMMANDS WITH OPTIONAL INTEGER VALUES PRECEDING THE COMMAND. THE EDITOR ACCEPTS BOTH UPPER AND LOWER CASE COMMANDS AND VALUES, AND PERFORMS TRANSLATION TO UPPER CASE UNDER THE FOLLOWING CONDITIONS. IF THE COMMAND IS TYPED IN UPPER CASE, THEN THE DATA WHICH

FOLLOWS IS TRANSLATED TO UPPER CASE. THUS, IF THE 'I' COMMAND IS TYPED IN UPPER CASE, THEN ALL INPUT IS AUTOMATICALLY TRANSLATED (ALTHOUGH ECHOED IN LOWER CASE, AS TYPED). IF THE 'A' COMMAND IS TYPED IN UPPER CASE, THEN ALL INPUT IS TRANSLATED AS READ FROM THE DISK. GLOBAL TRANSLATION TO UPPER CASE CAN BE CONTROLLED BY THE 'U' COMMAND (-U TO NEGATE ITS EFFECT). IF YOU ARE OPERATING WITH AN UPPER CASE ONLY TERMINAL, THEN OPERATION IS AUTOMATIC. SIMILARLY, IF YOU ARE OPERATING WITH A LOWER CASE TERMINAL, AND TRANSLATION TO UPPER CASE IS NOT SPECIFIED, THEN LOWER CASE CHARACTERS CAN BE ENTERED.

A NUMBER OF COMMANDS CAN BE PRECEDED BY A POSITIVE OR NEGATIVE INTEGER BETWEEN 0 AND 65535 (1 IS DEFAULT IF NO VALUE IS SPECIFIED). THIS VALUE DETERMINES THE NUMBER OF TIMES THE COMMAND IS APPLIED BEFORE RETURNING FOR ANOTHER COMMAND.

THE COMMANDS

C D K L T P U <CR>

CAN BE PRECEDED BY AN UNSIGNED, POSITIVE, OR NEGATIVE NUMBER, THE COMMANDS

A F J N W Z

CAN BE PRECEDED BY AN UNSIGNED OR POSITIVE NUMBER, THE COMMANDS

E H O Q

CANNOT BE PRECEDED BY A NUMBER. THE COMMANDS

F I J M R S

ARE ALL FOLLOWED BY ONE OR MORE STRINGS OF CHARACTERS WHICH CAN BE OPTIONALLY SEPARATED OR TERMINATED BY EITHER <ENDFILE> OR <CR>. THE <ENDFILE> IS GENERALLY USED TO SEPARATE THE SEARCH STRINGS IN THE S AND J COMMANDS, AND IS USED AT THE END OF THE COMMANDS IF ADDITIONAL COMMANDS FOLLOW. FOR EXAMPLE, THE FOLLOWING COMMAND SEQUENCE SEARCHES FOR THE STRING 'GAMMA', SUBSTITUTES THE STRING 'DELTA', AND THEN TYPES THE FIRST PART OF THE LINE WHERE THE CHANGE OCCURRED, FOLLOWED BY THE REMAINDER OF THE LINE WHICH WAS CHANGED:

SGAMMA<ENDFILE>DELTA<ENDFILE>OTT<CR>

THE CONTROL-L CHARACTER IN SEARCH AND SUBSTITUTE STRINGS IS REPLACED ON INPUT BY <CR><LF> CHARACTERS. THE CONTROL-I KEY IS TAKEN AS A TAB CHARACTER.

THE COMMANDS R & X MUST BE FOLLOWED BY A FILE NAME (WITH default FILE TYPE OF 'LIB') WITH A TRAILING <CR> OR <ENDFILE>. THE COMMAND I IS FOLLOWED BY A STRING OF SYMBOLS TO INSERT, TERMINATED BY A <CR> OR <ENDFILE>. IF SEVERAL LINES OF TEXT ARE TO BE INSERTED, THE I CAN BE DIRECTLY FOLLOWED BY AN <ENDFILE> OR <CR> IN WHICH CASE THE EDITOR ACCEPTS LINES OF INPUT TO THE NEXT <ENDFILE>. THE COMMAND OT PRINTS THE FIRST PART OF THE CURRENT LINE, AND THE COMMAND OL MOVES THE REFERENCE TO THE BEGINNING OF THE CURRENT LINE. THE COMMAND OP PRINTS THE CURRENT PAGE ONLY, WHILE THE COMMAND OZ READS THE CONSOLE RATHER THAN WAITING (THIS IS USED AGAIN WITHIN MACROS TO STOP THE DISPLAY - THE MACRO EXPANSION STOPS UNTIL A CHARACTER IS READ. IF THE CHARACTER IS NOT A BREAK THEN THE MACRO EXPANSION CONTINUES NORMALLY).

NOTE THAT A POUND SIGN IS TAKEN AS THE NUMBER 65535, ALL UNSIGNED NUMBERS ARE ASSUMED POSITIVE, AND A SINGLE - IS ASSUMED -1

A NUMBER OF COMMANDS CAN BE GROUPED TOGETHER AND EXECUTED REPETITIVELY USING THE MACRO COMMAND WHICH TAKES THE FORM

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

<NUMBER>MC1C2...CN<DELIMITER>

WHERE <NUMBER> IS A NON-NEGATIVE INTEGER N, AND <DELIMITER> IS <ENDFILE> OR <CR>. THE COMMANDS C1 ... CN FOLLOWING THE M ARE EXECUTED N TIMES, STARTING AT THE CURRENT POSITION IN THE BUFFER. IF N IS 0, 1, OR OMITTED, THE COMMANDS ARE EXECUTED UNTIL THE END OF THE BUFFER IS ENCOUNTERED.

THE FOLLOWING MACRO, FOR EXAMPLE, CHANGES ALL OCCURRENCES OF THE NAME 'GAMMA' TO 'DELTA', AND PRINTS THE LINES WHICH WERE CHANGED:

MFGAMMA<ENDFILE>-SDIDELTA<ENDFILE>OLT<CR>

(NOTE: AN <ENDFILE> IS THE CP/M END OF FILE MARK - CONTROL-Z)

IF ANY KEY IS DEPRESSED DURING TYPING OR MACRO EXPANSION, THE FUNCTION IS CONSIDERED TERMINATED, AND CONTROL RETURNS TO THE OPERATOR.

ERROR CONDITIONS ARE INDICATED BY PRINTING ONE OF THE CHARACTERS:

SYMBOL	ERROR CONDITION
GREATER	FREE MEMORY IS EXHAUSTED - ANY COMMAND CAN BE ISSUED WHICH DOES NOT INCREASE MEMORY REQUIREMENTS.
QUESTION	UNRECOGNIZED COMMAND OR ILLEGAL NUMERIC FIELD
POUND	CANNOT APPLY THE COMMAND THE NUMBER OF TIMES SPECIFIED (OCCURS IF SEARCH STRING CANNOT BE FOUND)
LETTER O	CANNOT OPEN <FILENAME>.LIB IN R COMMAND

THE ERROR CHARACTER IS ALSO ACCOMPANIED BY THE LAST CHARACTER SCANNED WHEN THE ERROR OCCURRED. */

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
$ eject
```

```
/* **** */
```

```
*** GLOBAL VARIABLES ***
```

```
***** */
```

```
30 1 DECLARE LIT LITERALLY 'LITERALLY',
    DCL LIT 'DECLARE',
    PROC LIT 'PROCEDURE',
    ADDR LIT 'ADDRESS',
    CTLL LIT '0CH',
    CTRR LIT '12H', /* REPEAT LINE IN INSERT MODE */
    CTLU LIT '15H', /* LINE DELETE IN INSERT MODE */
    CTXX LIT '18H', /* EQUIVALENT TO CTLU */
    CTRH LIT '08H', /* BACKSPACE */
    TAB LIT '09H', /* TAB CHARACTER */
    LCA LIT '110$0001B', /* LOWER CASE A */
    LCZ LIT '111$1010B', /* LOWER CASE Z */
    ESC LIT '1BH', /* ESCAPE CHARACTER */
    ENDFILE LIT '1AH'; /* CP/M END OF FILE */
```

```
31 1 DECLARE
    TRUE LITERALLY '1',
    FALSE LITERALLY '0',
    FOREVER LITERALLY 'WHILE TRUE',
    CTRL$C LITERALLY '3',
    CR LITERALLY '13',
    LF LITERALLY '10',
    WHAT LITERALLY '63';
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

```
32 1 DECLARE
    MAX ADDRESS, /* MEMORY(MAX)=0 (END) */
    MAXM ADDRESS, /* MINUS 1 */
    HMAX ADDRESS; /* = MAX/2 */
```

```
33 1 declare
    i byte; /* used by command parsing */
```

```
34 1 DECLARE
    us literally '8', /* file from user 0 */
    RO LITERALLY '9', /* R/O FILE INDICATOR */
    SY LITERALLY '10', /* SYSTEM FILE ATTRIBUTE */
    EX LITERALLY '12', /* EXTENT NUMBER POSITION */
    UB LITERALLY '13', /* UNFILLED BYTES */
    ck LITERALLY '13', /* checksum */
    MD LITERALLY '14', /* MODULE NUMBER POSITION */
    NR LITERALLY '32', /* NEXT RECORD FIELD */
    FS LITERALLY '33', /* FCB SIZE */
    RFCB (FS) BYTE /* READER FILE CONTROL BLOCK */
    INITIAL(0, /* FILE NAME */ ' ',
    /* FILE TYPE */ 'LIB', 0, 0, 0),
    RBP BYTE, /* READ BUFFER POINTER */
    XFCB (FS) BYTE /* XFER FILE CONTROL BLOCK */
```

```

        INITIAL(0, 'X$$$$$$', 'LIB', 0, 0, 0, 0, 0, 0),
XFCBE BYTE AT(.XFCB(EX)),      /* XFCB EXTENT */
XFCBM BYTE AT(.XFCB(MD)),      /* MODULE NUMBER */
XFCBR BYTE AT(.XFCB(NR)),      /* XFCB RECORD # */
xfcbe BYTE AT(.XFCB(ck)),      /* xfcbe check sum # */
xfcbebt byte initial(0),       /* save xfcbe extent for appends */
xfcbrrec byte initial(0),      /* save xfcbe record for appends */
XBUFF (SECTSIZE) BYTE,        /* XFER BUFFER */
XBP BYTE,                      /* XFER POINTER */
XFERON BYTE initial (false),   /* TRUE IF XFER ACTIVE */
reading BYTE initial (false), /* TRUE IF reading RFCB */

NBUF BYTE,                    /* NUMBER OF BUFFERS */
BUFFLENGTH ADDRESS,          /* NBUF * SECTSIZE */
SFCB (FS) BYTE AT(.FCB),      /* SOURCE FCB = DEFAULT FCB */
SDISK BYTE AT (.FCB),         /* SOURCE DISK */
SBUFFADR ADDRESS,             /* SOURCE BUFFER ADDRESS */
SBUFF BASED SBUFFADR (128) BYTE, /* SOURCE BUFFER */
password (16) byte initial(0), /* source password */

DFCB (FS) BYTE,               /* DEST FILE CONTROL BLOCK */
DDISK BYTE AT (.DFCB),        /* DESTINATION DISK */
DFUB BYTE AT(.DFCB(UB)),      /* UNFILLED BYTES IN LAST RECORD */
DBUFFADR ADDRESS,             /* DESTINATION BUFFER ADDRESS */
DBUFF BASED DBUFFADR (128) BYTE, /* DEST BUFFER */
NSOURCE ADDRESS,              /* NEXT SOURCE CHARACTER */
NDEST ADDRESS;                /* NEXT DESTINATION CHAR */

```

```

35 1  DECLARE
      newfile byte initial (false), /* no source file */
      onefile byte initial (true),  /* output file = input file */
      dtype (3) byte,               /* destination file type */
      libfcb (12) byte initial(0, 'X$$$$$$LIB'), /* default lib name */
      tempfl (3) byte initial('$$$'), /* temporary file type */
      backup (3) byte initial('BAK'); /* backup file type */

36 1  DECLARE
      PRINTSUPPRESS BYTE initial (false), /* TRUE IF PRINT SUPPRESSED */
      sys byte initial(0),                 /* true if system file */
      protection byte initial(0); /* password protection mode */

37 1  declare
      error$code address;

38 1  DECLARE
      COLUMN BYTE,                  /* CONSOLE COLUMN POSITION */
      SCOLUMN BYTE INITIAL(8),      /* STARTING COLUMN IN "I" MODE */
      TCOLUMN BYTE,                 /* TEMP DURING BACKSPACE */
      OCOLUMN BYTE;                 /* TEMP DURING BACKSPACE */

39 1  DECLARE DCNT BYTE;

```

```

/* COMMAND BUFFER */

```

COPYRIGHT 1981 1982
 DIGITAL RESEARCH
 P.O. Box 570
 PACIFIC GROVE, CA 93950

SER # _____

```

40 1  DECLARE (MAXLEN,COMLEN) BYTE, COMBUFF(128) BYTE,
      (TCBP,CBP) BYTE;

      /* MP/M PARSE FUNCTION CALL */
41 1  declare
      parse$fn structure (
          buff$adr address,
          fcb$a . address);

42 1  DECLARE /* LINE COUNTERS */
      BASELINE ADDRESS, /* CURRENT LINE */
      RELLINE ADDRESS, /* RELATIVE LINE IN TYPEDOUT */
      LINESET BYTE initial (true); /* TRUE IF LINE #'S PRINTED */

43 1  DECLARE
      LPP LIT '23', /* LINES PER PAGE */
      FORWARD LIT '1',
      BACKWARD LIT '0',
      RUBOUT LIT '07FH',
      POUND LIT '23H',
      MACSIZE LIT '128', /* MAX MACRO SIZE */
      SCRSIZE LIT '100', /* SCRATCH BUFFER SIZE */
      CONSIZE LIT 'ADDRESS'; /* DETERMINES MAX COMMAND NUMBER*/

44 1  DCL MACRO(MACSIZE) BYTE,
      SCRATCH(SCRSIZE) BYTE, /* SCRATCH BUFFER FOR F,N,S */
      (WBP, WBE, WBJ) BYTE, /* END OF F STRING, S STRING, J STRING */
      (FLAG, MP, MI, XP) BYTE,
      NT CONSIZE;

45 1  DCL (START, RESTART, OVERCOUNT, OVERFLOW,
      disk$full$err, dir$full$err, RESET, BADCOM) LABEL;

46 1  DCL INSERTING BYTE, /* TRUE IF INSERTING CHARACTERS */
      READBUFF BYTE; /* TRUE IF END OF READ BUFFER */

      /* global variables used by file parsing routines */
47 1  dcl nc$md byte initial(0),
      command$tail byte initial(0ffh);

48 1  DECLARE
      EOS LITERALLY '0FFH';

49 1  DCL (DISTANCE, TDIST) CONSIZE,
      (DIRECTION, CHAR) BYTE,
      ( FRONT, BACK, FIRST, LASTC) ADDR;

50 1  DECLARE
      TRANSLATE BYTE initial (false); /* TRUE IF TRANSLATION TO UPPER CASE */
      UPPER BYTE initial (false); /* TRUE IF GLOBALLY TRANSLATING TO UC */

51 1  declare ver address; /* VERSION NUMBER */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```
52 1  declare
      errmsg      address initial(0),
      invalid (*)  byte data ('Invalid Filename$'),
      dirfull (*)  byte data ('DIRECTORY FULL$'),
      diskfull (*) byte data ('DISK FULL$');
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

\$ eject

*** CP/M INTERFACE ROUTINES ***

/* IO SECTION */

```

53 1  READCHAR: PROCEDURE BYTE; RETURN MON2(1,0);
55 2      END READCHAR;

56 1      conin:
           procedure byte;
57 2      return mon2(6,0fdh);
58 2      end conin;

59 1  PRINTCHAR: PROCEDURE(CHAR);
60 2      DECLARE CHAR BYTE;
61 2      IF PRINTSUPPRESS THEN RETURN;
63 2      CALL MON1(2,CHAR);
64 2      END PRINTCHAR;

65 1  TTYCHAR: PROCEDURE(CHAR);
66 2      DECLARE CHAR BYTE;
67 2      IF CHAR >= ' ' THEN COLUMN = COLUMN + 1;
69 2      IF CHAR = LF THEN COLUMN = 0;
71 2      CALL PRINTCHAR(CHAR);
72 2      END TTYCHAR;

73 1  BACKSPACE: PROCEDURE;
           /* MOVE BACK ONE POSITION */
74 2      IF COLUMN = 0 THEN RETURN;
76 2      CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
77 2      CALL TTYCHAR(' '); /* COLUMN = COLUMN + 1 */
78 2      CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
79 2      COLUMN = COLUMN - 2;
80 2      END BACKSPACE;

81 1  PRINTABS: PROCEDURE(CHAR);
82 2      DECLARE (CHAR,I,J) BYTE;
83 2      I = CHAR = TAB AND 7 - (COLUMN AND 7);
84 2      IF CHAR = TAB THEN CHAR = ' ';
86 2      DO J = 0 TO I;
87 3          CALL TTYCHAR(CHAR);
88 3      END;
89 2      END PRINTABS;

90 1  GRAPHIC: PROCEDURE(C) BYTE;
91 2      DECLARE C BYTE;
           /* RETURN TRUE IF GRAPHIC CHARACTER */
92 2      IF C >= ' ' THEN RETURN TRUE;
94 2      RETURN C = CR OR C = LF OR C = TAB;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```
95 2      END GRAPHIC;

96 1      PRINTC: PROCEDURE(C);
97 2      DECLARE C BYTE;
98 2      IF NOT GRAPHIC(C) THEN
99 2          DO; CALL PRINTABS('A');
101 3          C = C + '0';
102 3          END;
103 2      CALL PRINTABS(C);
104 2      END PRINTC;

105 1      CRLF: PROCEDURE;
106 2      CALL PRINTC(CR); CALL PRINTC(LF);
108 2      END CRLF;

109 1      PRINTN: PROCEDURE(A);
110 2      DECLARE A ADDRESS;
111 2      CALL MON1(9,A);
112 2      END PRINTN;

113 1      PRINT: PROCEDURE(A);
114 2      DECLARE A ADDRESS;
115 2      CALL CRLF;
116 2      CALL PRINTN(A);
117 2      END PRINT;

118 1      READ: PROCEDURE(A);
119 2      DECLARE A ADDRESS;
120 2      CALL MON1(10,A);
121 2      END READ;

/* used for library files */
122 1      OPEN: PROCEDURE(FCB);
123 2      DECLARE FCB ADDRESS;
124 2      IF MON2(15,FCB) = 255 THEN DO;
126 3          FLAG = '0';
127 3          GO TO RESET;
128 3          END;
129 2      END OPEN;

/* used for main source file */
130 1      OPEN$FILE: PROCEDURE(FCB) ADDRESS;
131 2      DECLARE FCB ADDRESS;
132 2      RETURN MON3(15,FCB);
133 2      END OPEN$FILE;

134 1      CLOSE: PROCEDURE(FCB);
135 2      DECLARE FCB ADDRESS;
136 2      DCNT = MON2(16,FCB);
137 2      END CLOSE;

138 1      SEARCH: PROCEDURE(FCB);
139 2      DECLARE FCB ADDRESS;
140 2      DCNT = MON2(17,FCB);
141 2      END SEARCH;

142 1      DELETE: PROCEDURE(FCB);
```

CP/M-86 v.1.1 (Vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # 001-162

```
143 2    DECLARE FCB ADDRESS;
144 2    DCNT = MON2(19,FCB);
145 2    END DELETE;

146 1    DISKREAD: PROCEDURE(FCB) BYTE;
147 2    DECLARE FCB ADDRESS;
148 2    RETURN MON2(20,FCB);
149 2    END DISKREAD;

150 1    DISKWRITE: PROCEDURE(FCB) BYTE;
151 2    DECLARE FCB ADDRESS;
152 2    RETURN MON2(21,FCB);
153 2    END DISKWRITE;

154 1    MAKE: PROCEDURE(FCB);
155 2    DECLARE FCB ADDRESS;
156 2    DCNT = MON2(22,FCB);
157 2    END MAKE;

158 1    RENAME: PROCEDURE(FCB);
159 2    DECLARE FCB ADDRESS;
160 2    CALL MON1(23,FCB);
161 2    END RENAME;

162 1    READCOM: PROCEDURE;
163 2    MAXLEN = 128; CALL READ(.MAXLEN);
165 2    END READCOM;

166 1    BREAK$KEY: PROCEDURE BYTE;
167 2    IF MON2(11,0) THEN
168 2        DO; /* CLEAR CHAR */
169 3        IF MON2(1,0) = CTRL$C THEN
170 3            RETURN TRUE;
171 3        END;
172 2    RETURN FALSE;
173 2    END BREAK$KEY;

174 1    CSELECT: PROCEDURE BYTE;
175 2    /* RETURN CURRENT DRIVE NUMBER */
176 2    RETURN MON2(25,0);
177 2    END CSELECT;

177 1    SELECT: PROCEDURE(DISK);
178 2    DECLARE DISK BYTE;
179 2    /* SET DRIVE NUMBER */
180 2    CALL MON1(14,DISK);
181 2    END SELECT;

181 1    SETDMA: PROCEDURE(A);
182 2    DECLARE A ADDRESS;
183 2    /* SET DMA ADDRESS */
184 2    CALL MON1(26,A);
185 2    END SETDMA;

185 1    set$attribute: procedure(FCB);
186 2    declare fcb address;
187 2    call MON1(30,FCB);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

188 2      end set$attribute;

/* The PL/M built-in procedure "MOVE" can be used to move storage,
   its definition is:

MOVE: PROCEDURE(COUNT,SOURCE,DEST);
      DECLARE (COUNT,SOURCE,DEST) ADDRESS;
      / MOVE DATA FROM SOURCE TO DEST ADDRESSES, FOR COUNT BYTES /
      END MOVE;
*/

/* this routine is included solely for
   economy of space over the use of the
   equivalent (in-line) code generated by
   the built-in function */
189 1  move:  proc(c,s,d);
190 2      dcl (s,d) addr, c byte;
191 2      dcl a based s byte, b based d byte;

192 2      do while (c:=c-1)<>255;
193 3          b=a; s=s+1; d=d+1;
196 3          end;
197 2      end move;

198 1  write$xfcb: PROCEDURE(FCB);
199 2      DECLARE FCB ADDRESS;
200 2      call move(8,.password,.password(8));
201 2      if MON2(103,FCB)= 0ffh then
202 2          call print(,('Error Creating Password$'));
203 2      END write$xfcb;

204 1  read$xfcb: PROCEDURE(FCB);
205 2      DECLARE FCB ADDRESS;
206 2      call MON1(102,FCB);
207 2      END read$xfcb;

/* Off => return BDOS errors */
208 1  return$errors:
      procedure(mode);
209 2      declare mode byte;
210 2      call mon1 (45,mode);
211 2      end return$errors;

212 1  REBOOT: PROCEDURE;
213 2      IF XFERON THEN
214 2          CALL DELETE(.libfcb);
215 2          CALL BOOT;
216 2          END REBOOT;

217 1  version: procedure address;
      /* returns current cp/m version # */
218 2      return mon3(12,0);
219 2      end version;

220 1  parse: procedure;
221 2      call mon1(152,.parse$fn);
222 2      end parse;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```
$ eject
```

```
/* **** */
```

```
*** SUBROUTINES ***
```

```
***** */
```

```
/* INPUT / OUTPUT BUFFERING ROUTINES */
```

```
/* abort ED and print error message */
```

```
223 1  ABORT: PROCEDURE(A);
224 2    DECLARE A ADDRESS;
225 2    CALL PRINT(A);
226 2    CALL CRLF;
227 2    CALL REBOOT;
228 2    END ABORT;
```

```
/* ----- */
```

```
/* fatal file error */
```

```
229 1  FERR: PROCEDURE;
230 2    CALL CLOSE(.DFCB); /* ATTEMPT TO CLOSE FILE FOR LATER RECOVERY */
231 2    CALL ABORT (.dirfull);
232 2    END FERR;
```

```
/* ----- */
```

```
/* set password if cpm 3 */
```

```
233 1  setpassword: procedure;
234 2    if low(ver) = cpm3 then
235 2        call setdwa(.password);
236 2    end setpassword;
```

```
/* ----- */
```

```
/* delete file at afcb */
```

```
237 1  delete$file: procedure(afcb);
238 2    declare afcb address;
239 2    call setpassword;
240 2    call delete(afcb);
241 2    end delete$file;
```

```
/* ----- */
```

```
/* rename file at afcb */
```

```
242 1  rename$file: procedure(afcb);
243 2    declare afcb address;
244 2    call delete$file(afcb+16); /* delete new file */
245 2    call setpassword;
246 2    call rename(afcb);
247 2    end rename$file;
```

COPYRIGHT 1981, 1982

DIGITAL RESEARCH

P.O. BOX 379

PACIFIC GROVE, CA 92350

SER. =

```
/* ----- */  
  
/* make file at afcb */  
248 1 make$file: procedure(afcb);  
249 2   declare afcb address;  
250 2   call delete$file(afcb); /* delete file */  
251 2   call setpassword;  
252 2   call make(afcb);  
253 2   end make$file;  
/* ----- */
```

```
/* fill string @ s for c bytes with f */  
254 1 fill: proc(s,f,c);  
255 2   dcl s addr,  
      (f,c) byte,  
      a based s byte;  
  
256 2   do while (c:=c-1) > 255;  
257 3     a = f;  
258 3     s = st1;  
259 3     end;  
260 2   end fill;
```

COPYRIGHT © 1981, 1982 .

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. = _____


```
$ eject
```

```
/* ****
```

```
*** FILE HANDLING ROUTINES ***
```

```
*****/
```

```
/* set destination file type to type at A */
```

```
261 1 SETTYPE: PROCEDURE(afeb,A);
262 2   DECLARE (afeb, A) ADDRESS;
263 2   CALL MOVE(3,A,afeb+9);
264 2   END SETTYPE;
/* ----- */
```

```
/* set dma to xfer buffer */
```

```
265 1 SETDMA: PROCEDURE;
266 2   CALL SETDMA(XBUF);
267 2   END SETDMA;
/* ----- */
```

```
/* fill primary source buffer */
```

```
268 1 FILLSOURCE: PROCEDURE;
269 2   DECLARE I BYTE;
270 2   ZN: PROCEDURE;
271 3     NSOURCE = 0;
272 3     END ZN;

273 2   CALL ZN;
274 2   DO I = 0 TO NBUF;
275 3     CALL SETDMA(SBUFFADR+NSOURCE);
276 3     IF (DCNT := DISKREAD(.FCB)) <> 0 THEN
277 3       DO; IF DCNT > 1 THEN CALL FERR;
280 4       SBUFF(NSOURCE) = ENDFILE;
281 4       I = NBUF;
282 4       END;
283 3     ELSE
284 3       NSOURCE = NSOURCE + SECTSIZE;
285 2     CALL ZN;
286 2     END FILLSOURCE;
/* ----- */
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
/* get next character in source file */
```

```
287 1 GETSOURCE: PROCEDURE BYTE;
288 2   DECLARE B BYTE;
289 2   IF NSOURCE >= BUFLLENGTH THEN CALL FILLSOURCE;
291 2   IF (B := SBUFF(NSOURCE)) <> ENDFILE THEN
292 2     NSOURCE = NSOURCE + 1;
293 2   RETURN B;
```

```

294 2      END GETSOURCE;
/* ----- */

```

```

/* try to free space by erasing backup */
295 1      erase$bak: PROCEDURE BYTE;
296 2          DECLARE B BYTE;
297 2          if onefile then
298 2              if newfile then dr
300 3              CALL SETTYPE(.DFCB,.BACKUP);
301 3              CALL DELETE$file(.DFCB);
302 3              call settype(.dfcb,.tempfl);
303 3              if dcnt < 255 then
304 3                  return true;
305 3              end;
306 2          return false;
307 2          end erase$bak;

```

```

/* ----- */

```

```

/* write output buffer up to (not including)
   ndest (low 7 bits of ndest are 0 */
308 1      WRITEDEST: PROCEDURE;
309 2          DECLARE (I,N) BYTE;
310 2          ZN: PROCEDURE;
311 3              NDEST = 0;
312 3              END ZN;

313 2          IF LOW((N := SHR(NDEST,SECTSHF) - 1)) = 255 THEN RETURN;
315 2          CALL ZN;
316 2          DO I = 0 TO N;
317 3          retry:
318 3              CALL SETDMA(DBUFFADR+NDEST);
319 3              IF DISKWRITE(.DFCB) < 0 THEN
320 3                  if erase$bak then
321 3                      go to retry;
322 4                  else do;
323 4                      call close(.dfcb);
324 4                      go to disk$full$err;
325 3                  end;
326 3                  NDEST = NDEST + SECTSIZE;
327 2              END;
328 2              CALL ZN;
329 2          END WRITEDEST;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/* put a character in output buffer */
329 1      PUTDEST: PROCEDURE(B);
330 2          DECLARE B BYTE;
331 2          IF NDEST >= BUFFLENGTH THEN CALL WRITEDEST;
332 2          DBUFF(NDEST) = B;
333 2          NDEST = NDEST + 1;
334 2          END PUTDEST;
/* ----- */

```

```

/* put a character in the xfer buffer */
336 1  PUTXFER: PROCEDURE(C);
337 2      DECLARE C BYTE;
338 2      IF XBP >= SECTSIZE THEN /* BUFFER OVERFLOW */
339 2          DO;
340 3      retry:
          CALL SETXDMA;
341 3          xfcbe = xfcbe; /* save for appends */
342 3          xfcbr = xfcbr;
343 3          IF DISKWRITE(.XFCB) <> 0 THEN
344 3              if erase$bak then
345 3                  go to retry;
346 3              else do;
347 4                  call close(.xfcb);
348 4                  go to disk$full$err;
349 4              end;
350 3          XBP = 0;
351 3          END;
352 2      XBUFF(XBP) = C; XBP = XBP + 1;
354 2      END PUTXFER;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

/* empty xfer buffer and close file.
   This routine is added to allow saving lib
   files for future edits - DH 10/18/81 */
355 1  close$xfer: procedure;
356 2      dcl i byte;

357 2      do i = xbp to sectsize;
358 3          call putxfer(ENDFILE);
359 3          end;
360 2      call close(.xfcb);
361 2      end close$xfer;
/* ----- */

```

```

/* compare xfcf and rfcf to see if same */
362 1  compare$xfer: procedure byte;
363 2      dcl i byte;

364 2      i = 12;
365 2      do while (i:=i-1) <> -1;
366 3          if xfcf(i) <> rfcf(i) then
367 3              return false;
368 3          end;
369 2      return true;
370 2      end compare$xfer;
/* ----- */

```

```

/* restore xfer file extent and current
   record, read record and set xfer pointer
   to first ENDFILE */

```

```
372 2    xfcbe = xfcbe;
373 2    call open(.xfcb);
374 2    xfcbr = xfcbr;
375 2    call setxdma;
376 2    if diskread(.xfcb) = 0 then do;
378 3        xfcbr = xfcbr;          /* write same record */
379 3        do xbp = 0 to sectsize;
380 4            if xbuff(xbp) = ENDFILE then
381 4                return;
382 4        end;
383 3    end;
384 2    end append$fer;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

/* **** */

*** END EDIT ROUTINE ***

**** */

/* finish edit, close files, rename */

```

385 1  FINIS: PROCEDURE;
386 2      MOVEUP: PROCEDURE(afeb);
387 3      dcl afeb address;
          /* set second filename (new name) for rename function */
388 3      CALL MOVE(16,afeb,afeb+16);
389 3      END MOVEUP;

          /* **** WRITE OUTPUT BUFFER **** */

          /* SET UNFILLED BYTES - USED FOR ISIS-II COMPATIBILITY */
          /* DFUB = 0 ; <<<< REMOVE FOR HP/M 2 , CP/M 3 */
390 2      DO WHILE (LOW(NDEST) AND 7FH) <> 0;
          /* COUNTS UNFILLED BYTES IN LAST RECORD */
          /* DFUB = DFUB + 1; */
391 3      CALL PUTDEST(ENDFILE);
392 3      END;
393 2      CALL WRITEDEST;

          /* **** CLOSE TEMPORARY DESTINATION FILE **** */

394 2      CALL CLOSE(.DFCB);
395 2      IF DCNT = 255 THEN CALL FERR;
397 2      if sys then do;
399 3          dfcb(sy)=dfcb(sy) or 80h;
400 3          call setpassword;
401 3          call set$attribute(.dfcb);
402 3          end;

```

/* **** RENAME SOURCE TO BACKUP IF ONE FILE **** */

```

403 2      if onefile then do;
405 3          call moveup(.sfcb);
406 3          CALL SETTYPE(.sfcb+16,.BACKUP); /* set new type to BAK */
407 3          CALL RENAME$FILE(.SFEB);
408 3          end;

```

/* **** RENAME TEMPORARY DESTINATION FILE **** */

```

409 2      CALL MOVEUP(.DFCB);
410 2      CALL SETTYPE(.DFCB+16,.DTYPE);
411 2      CALL RENAME$FILE(.DFCB);

412 2      END FINIS;

```

COPYRIGHT © 1981, 1982

*****/ DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. =


```
$ eject
```

```
/* ****
```

```
*** COMMAND ROUTINES ***
```

```
*****~*****/
```

```
/* print a character if not macro expansion */
```

```
413 1 PRINTMAC: PROCEDURE(CHAR);
414 2   DECLARE CHAR BYTE;
415 2   IF MP < 0 THEN RETURN;
417 2   CALL PRINTC(CHAR);
418 2   END PRINTMAC;
```

```
/* ----- */
```

```
/* return true if lower case character */
```

```
419 1 LOWERCASE: PROCEDURE(C) BYTE;
420 2   DECLARE C BYTE;
421 2   RETURN C >= LCA AND C <= LCZ;
422 2   END LOWERCASE;
```

```
/* ----- */
```

```
/* translate character to upper case */
```

```
423 1 UCASE: PROCEDURE(C) BYTE;
424 2   DECLARE C BYTE;
425 2   IF LOWERCASE(C) THEN RETURN C AND 5FH;
427 2   RETURN C;
428 2   END UCASE;
```

```
/* ----- */
```

```
/* get password and place at fcb + 16 */
```

```
429 1 getpasswd: proc;
430 2   dcl (i,c) byte;

431 2   call crlf;
432 2   call print(,('Password ? ','$'));
433 2   retry:
      call fill(,password,' ',8);
434 2   do i = 0 to 7;
435 3   nxtchr:
      if (c:=ucase(conin)) >= ' ' then
436 3       password(i)=c;
437 3       if c = cr then
438 3         go to exit;
439 3       if c = CTLX then
440 3         goto retry;
441 3       if c = CTLH then do;
443 4         if i<1 then
444 4           goto retry;
```

CP/M-86 v.1.1 (Vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```

445 4          else do;
446 5              password(i:=i-1)=' ';
447 5              goto nxtchr;
448 5          end;
449 4          end;
450 3          if c = 3 then
451 3              call reboot;
452 3          end;
453 2      exit:
          c = break$key;      /* clear raw I/O mode */
454 2      end getpasswd;
/* ----- */

```

```

          /* translate to upercase if translate flag
          is on (also translate ESC to ENDFILE) */
455 1      UTRAN: PROCEDURE(C) BYTE;
456 2          DECLARE C BYTE;
457 2          IF C = ESC THEN C = ENDFILE;
459 2          IF TRANSLATE THEN RETURN UCASE(C);
461 2          RETURN C;
462 2      END UTRAN;
/* ----- */

```

```

          /* print the line number */
463 1      PRINTVALUE: PROCEDURE(V);
          /* PRINT THE LINE VALUE V */
464 2          DECLARE (D,ZERO) BYTE,
          (K,V) ADDRESS;
465 2          K = 10000;
466 2          ZERO = FALSE;
467 2          DO WHILE K <> 0;
468 3              D = LOW(V/K); V = V MOD K;
470 3              K = K / 10;
471 3              IF ZERO OR D <> 0 THEN
472 3                  DO; ZERO = TRUE;
474 4                  CALL PRINTC('0'+D);
475 4                  END;
          ELSE
476 3              CALL PRINTC(' ');
477 3          END;
478 2      END PRINTVALUE;
/* ----- */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

          /* print line with number V */
479 1      PRINTLINE: PROCEDURE(V);
480 2          DECLARE V ADDRESS;
481 2          IF NOT LINESET THEN RETURN;
483 2          CALL PRINTVALUE(V);
484 2          CALL PRINTC(' ');
485 2          CALL PRINTC(' ');
486 2          IF INSERTING THEN
487 2              CALL PRINTC(' ');
          ELSE
488 2              CALL PRINTC('*');

```



```

489 2      END PRINTLINE;
/* ----- */

/* print current line (baseline) */
490 1      PRINTBASE: PROCEDURE;
491 2          CALL PRINTLINE(BASELINE);
492 2      END PRINTBASE;
/* ----- */

/* print current line if not in a macro */
493 1      PRINTNMBASE: PROCEDURE;
494 2          IF MP <> 0 THEN RETURN;
496 2          CALL PRINTBASE;
497 2      END PRINTNMBASE;
/* ----- */

/* get next character from command tail */
498 1      getcmd: proc byte;

499 2          if buff(ncmd + 1) <> 0 then
500 2              return buff(ncmd := ncmd + 1);
          else
501 2              return CR;
502 2          end getcmd;
/* ----- */

/* read next char from command buffer */
503 1      READC: PROCEDURE BYTE;
/* MAY BE MACRO EXPANSION */
504 2          IF MP > 0 THEN
505 2              DO;
506 3              IF BREAK$KEY THEN GO TO OVERCOUNT;
508 3              IF XP >= MP THEN
509 3                  DO; /* START AGAIN */
510 4                  IF MT <> 0 THEN
511 4                      DO; IF (MT:=MT-1) = 0 THEN
513 5                          GO TO OVERCOUNT;
514 5                      END;
515 4                      XP = 0;
516 4                      END;
517 3                      RETURN UTRAN(MACRO((XP := XP + 1) - 1));
518 3                      END;
519 2          IF INSERTING THEN RETURN UTRAN(READCHAR);

/* GET COMMAND LINE */
521 2          IF READBUFF THEN
522 2              DO; READBUFF = FALSE;
524 3              IF LINESET AND COLUMN = 0 THEN
525 3                  DO;
526 4                  IF BACK >= MAXM THEN
527 4                      CALL PRINTLINE(0);
                    ELSE
528 4                      CALL PRINTBASE;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

529 4          END;
          ELSE
530 3          CALL PRINTC('*');
531 3          CALL READCOM; CBP = 0;
533 3          CALL PRINTC(LF);
534 3          COLUMN = 0;
535 3          END;
536 2          IF (READBUFF := CBP = COMLEN ) THEN
537 2              COMBUFF(CBP) = CR;
538 2          RETURN UTRAN(COMBUFF((CBP := CBP +1) -1));
539 2          END READC;
/* ----- */

```

```

          /* get upper case character from command
          buffer or command line */
540 1  get$uc: proc;
541 2      if command$tail then
542 2          char = ucase(getcmd);
          else
543 2          char = ucase(readc);
544 2      end get$uc;
/* ----- */

```

```

          /* parse file name
          this routine requires a routine to get
          the next character and put it in a byte
          variable */
545 1  parse$fcb: proc(fcbadr) byte;
546 2      dcl fcbadr addr;
547 2      dcl afcb based fcbadr (33) byte;
548 2      dcl drive lit 'afcb(0)';
549 2      dcl (i,flag,delimiter) byte;

```

```

550 2      putc: proc;
551 3          afcb(i := i + 1) = char;
552 3          flag = true;
553 3      end putc;

```

```

554 2      delim: proc byte;
555 3          dcl del(*) byte data (CR,ENDFILE,' ,.=:<>_[]*?');
          /* 0      1 234567890123 */
556 3          do delimiter = 0 to last(del);
557 4              if char = '?' or char = '*' then
558 4                  call abort(.(('Cannot Edit Wildcard Filename$'));
559 4              if char = del(delimiter) then
560 4                  return (true);
561 4              end;
562 3          return (false);
563 3      end delim;

```

```

564 2      flag = false;
565 2      call get$uc;
566 2      if char <> CR then

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

567 2      if char <> ENDFILE then do;
           /* initialize fcb to srce fcb type & drive */
569 3      call fill(fcbadr+12,0,21);
570 3      call fill(fcbadr+1,' ',11);
           /* clear leading blanks */
571 3      do while char = ' ';
572 4      call get$uc;
573 4      end;
           /* parse loop */
574 3      do while not delim;
575 4      i = 0;
           /* get name */
576 4      do while not delim;
577 5      if i > 8 then
578 5      go to err;      /* too long */
579 5      call putc;
580 5      call get$uc;
581 5      end;
582 4      if char = ':' then do;
           /* get drive from afcb(1) */
584 5      if i <> 1 then
585 5      go to err;      /* invalid : */
586 5      if (drive := afcb(1) - 'A' + 1) > 16 then
587 5      go to err;      /* invalid drive */
588 5      afcb(1) = ' ';
589 5      call get$uc;
590 5      end;
591 4      if char = '.' then do;
           /* get file type */
593 5      i = 8;
594 5      call get$uc;
595 5      do while not delim;
596 6      if i > 11 then
597 6      go to err;      /* too long */
598 6      call putc;
599 6      call get$uc;
600 6      end;
601 5      end;
602 4      end;      /* parse loop */
           /* delimiter must be a comma or space */
603 3      if delimiter > 3 then /* not a CR,ENDFILE,SPACE,COMMA */
604 3      go to err;
605 3      if not flag then
606 3      go to err;
607 3      end;

608 2      return (flag);

609 2      err:
610 2      call abort(.invalid);
611 2      return (false);
           end parse$fcb;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

613 2      CALL MOVE(16,.FCB,.DFCB);
614 2      END copydest;
/* ----- */

```

```

/* set up destination FCB */
615 1  setdest: PROCEDURE;
616 2      dcl i byte;

/* onefile = true; (initialized) */
617 2      if parse$fcbl(.dfcb) then
618 2          do i=1 to 11;
619 3          if fcb(i) <> dfcb(i) then
620 3              if dfcb(1) <> ' ' then
621 3                  onefile = false;
622 3          end;
623 2      if onefile then
624 2          call copydest;
625 2      call move(3,.dfcb(9),.dtype); /* save destination type */
626 2      if getcmd <> CR then
627 2          call abort(.invalid);
628 2      end setdest;
/* ----- */

```

```

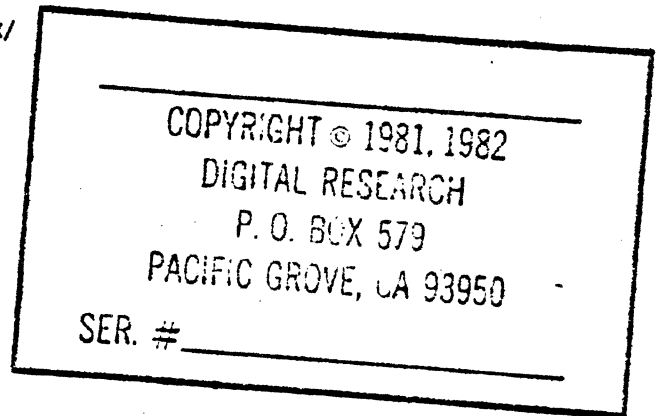
/* set read lib file DMA address */
629 1  SETRDMA: PROCEDURE;
630 2      CALL SETDMA(.BUFF);
631 2      END SETRDMA;
/* ----- */

```

```

/* read lib file routine */
632 1  READFILE: PROCEDURE BYTE;
633 2      IF RBP >= SECTSIZE THEN
634 2          DO; CALL SETRDMA;
635 3          IF DISKREAD(.RFCB) <> 0 THEN RETURN ENDFILE;
636 3          RBP = 0;
637 3          END;
638 2      RETURN UTRAN(BUFF((RBP := RBP + 1) - 1));
639 2      END READFILE;

```



```
$ eject
```

```
/* **** */
```

```
*** INITIALIZATION ***
```

```
/* **** */
```

```
642 1  SETUP: PROCEDURE;
```

```
/* **** OPEN SOURCE FILE **** */
```

```
643 2      sfcb(ex), sfcb(md), sfcb(nr) = 0;
644 2      if low(ver)=cpm3 then do;
646 3          call return$errors(OFeh);          /* set error mode */
647 3          call setpassword;
648 3          end;
649 2      error$code = open$file (.fcb);
650 2      if low(ver)=cpm3 then do;
652 3          call return$errors(0);          /* reset error mode */
653 3          if low(error$code) = OFFh then
654 3              if high(error$code) = 7 then do;
656 4              call getpasswd;
657 4              call crlf;
658 4              call crlf;
659 4              call setpassword;          /* set dma to password */
660 4              error$code = open$file(.fcb);
661 4              end;
662 3          end;
663 2      dcnt=low(error$code);
664 2      if onefile then do;
666 3          IF ROL(FCB(KO),1) THEN
667 3              CALL abort(.(('FILE IS READ/ONLY$')));
668 3          else IF ROL(FCB(SY),1) THEN /* system attribute */
669 3              do;
670 4              if rol(FCB(us),1) then
671 4                  dcnt = 255;          /* user 0 file so create */
672 4              else
673 4                  sys = true;
674 4              end;
675 2          end;
```

```
/* **** NEW FILE IF NO SOURCE FILE **** */
```

```
675 2      IF DCNT = 255 THEN do;
677 3          if not onefile then
678 3              call abort(.(('File not Found$')));
679 3          newfile = true;
680 3          CALL PRINT(.(('NEW FILE$')));
681 3          CALL CRLF;
682 3          END;
```

```
/* **** MAKE TEMPORARY DESTINATION FILE **** */
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 879

PACIFIC GROVE, CA 93950

SEP 16

```
683 2    CALL SETTYPE(.dfcb,.tempfl);
684 2    DFCB(EX)=0;
685 2    CALL MAKE$file(.DFCB);
686 2    if dcnt = 255 then
687 2        call ferr;
/* THE TEMP FILE IS NOW CREATED */

/* now create the password if any */
688 2    if protection < ( then do;
689 3        dfcb(ex) = protection or 1; /* set password */
690 3        call setpassword;
691 3        call write$xfcb(.dfcb);
692 3        end;
693 3    dfcb(ex),DFCB(32) = 0; /* NEXT RECORD IS ZERO */
694 2

/* * * * * * RESET BUFFER * * * * * */

695 2    NSOURCE = BUFFLENGTH;
696 2    NDEST = 0;
697 2    BASELINE = 1; /* START WITH LINE 1 */
698 2    END SETUP;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93350
SER. # _____

```
$ eject
```

```
/* ****
```

```
*** BUFFER MANAGEMENT ***
```

```
***** */
```

```
/* DISTANCE is the number of lines prefix  
to a command */
```

```
/* set maximum distance (OFFFH) */
```

```
699 1 SETFF: PROCEDURE;  
700 2     DISTANCE = OFFFH;  
701 2     END SETFF;  
/* ----- */
```

```
/* return true if distance is zero */
```

```
702 1 DISTZERO: PROCEDURE BYTE;  
703 2     RETURN DISTANCE = 0;  
704 2     END DISTZERO;  
/* ----- */
```

```
/* set distance to zero */
```

```
705 1 ZERODIST: PROCEDURE;  
706 2     DISTANCE = 0;  
707 2     END ZERODIST;  
/* ----- */
```

```
/* check for zero distance and decrement */
```

```
708 1 DISTNZERO: PROCEDURE BYTE;  
709 2     IF NOT DISTZERO THEN  
710 2         DO; DISTANCE = DISTANCE - 1;  
712 3         RETURN TRUE;  
713 3         END;  
714 2     RETURN FALSE;  
715 2     END DISTNZERO;  
/* ----- */
```

```
/* set memory limits of command from  
distance and direction */
```

```
716 1 SETLIMITS: PROC;  
717 2     DCL (I,K,L,K) ADDR, (MIDDLE,LOOPING) BYTE;  
718 2     RELLINE = 1; /* RELATIVE LINE COUNT */  
719 2     IF DIRECTION = BACKWARD THEN  
720 2         DO; DISTANCE = DISTANCE+1; I = FRONT; L = 0; K = OFFFH;  
725 3         END;  
726 2     ELSE  
727 2         DO; I = BACK; L = MAXH; K = 1;  
728 2         END;  
729 2     END SETLIMITS;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. ==

```

731 2      LOOPING = TRUE;
732 2      DO WHILE LOOPING;
733 3          DO WHILE (MIDDLE := I < L) AND
              MEMORY(M:=I+K) < LF;
734 4              I = M;
735 4              END;
736 3          LOOPING = (DISTANCE := DISTANCE - 1) < 0;
737 3          IF NOT MIDDLE THEN
738 3              DO; LOOPING = FALSE;
740 4              I = I - K;
741 4              END;
742 3          ELSE do;
743 4              RELLINE = RELLINE - 1;
744 4              IF LOOPING THEN
745 4                  I = M;
746 4              end;
747 3          END;

748 2      IF DIRECTION = BACKWARD THEN
749 2          DO; FIRST = I; LASTC = FRONT - 1;
752 3          END;
753 2      ELSE
754 2          DO; FIRST = BACK + 1; LASTC = I + 1;
755 3          END;
757 2      END SETLIMITS;

/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

/* increment current position */
758 1  INCBASE: PROCEDURE;
759 2      BASELINE = BASELINE + 1;
760 2      END INCBASE;

/* ----- */

```

```

/* decrement current position */
761 1  DECBASE: PROCEDURE;
762 2      BASELINE = BASELINE - 1;
763 2      END DECBASE;

/* ----- */

```

```

/* increment limits */
764 1  INCFRONT: PROC; FRONT = FRONT + 1;
766 2      END INCFRONT;
767 1  INCBACK: PROCEDURE; BACK = BACK + 1;
769 2      END INCBACK;

/* ----- */

```

```

/* decrement limits */
770 1  DECFRONT: PROC; FRONT = FRONT - 1;
772 2      IF MEMORY(FRONT) = LF THEN
773 2          CALL DECBASE;
774 2      END DECFRONT;
775 1  DECBACK: PROC; BACK = BACK - 1;

```



```

777 2      END DECBACK;
/* ----- */

/* move current page in memory if move flag
   true otherwise delete it */
778 1  MEM$MOVE: PROC(MOVEFLAG);
779 2      DECLARE (MOVEFLAG,C) BYTE;
/* MOVE IF MOVEFLAG IS TRUE */
780 2      IF DIRECTION = FORWARD THEN
781 2          DO WHILE BACK < LASTC; CALL INCBACK;
783 3          IF MOVEFLAG THEN
784 3              DO;
785 4              IF (C := MEMORY(BACK)) = LF THEN CALL INCBASE;
787 4              MEMORY(FRONT) = C; CALL INCFRONT;
789 4              END;
790 3          END;
      ELSE
791 2          DO WHILE FRONT > FIRST; CALL DECFRONT;
793 3          IF MOVEFLAG THEN
794 3              DO; MEMORY(BACK) = memory(front); CALL DECBACK;
797 4              END;
798 3          END;
799 2      END MEM$MOVE;
/* ----- */

/* force a memory move */
800 1  MOVER: PROC;
801 2      CALL MEM$MOVE(TRUE);
802 2      END MOVER;
/* ----- */

/* reset memory limit pointers, deleting
   characters (used by D command) */
803 1  SETPTRS: PROC;
804 2      CALL MEM$MOVE(FALSE);
805 2      END SETPTRS;
/* ----- */

/* set limits and force a move */
806 1  NOVELINES: PROC;
807 2      CALL SETLIMITS;
808 2      CALL MOVER;
809 2      END NOVELINES;
/* ----- */

/* set front to lower value deleting
   characters (used by S and J commands) */
810 1  setfront: proc(newfront);
811 2      dcl newfront addr;

812 2      do while front > newfront;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

814 3      end;
815 2      end setfront;
/* ----- */

```

```

/* set limits for memory move */
816 1      SETCLIMITS: PROC;
817 2          IF DIRECTION = BACKWARD THEN
818 2              DO; LASTC = BACK;
820 3              IF DISTANCE > FRONT THEN
821 3                  FIRST = 1;
                ELSE
822 3                  FIRST = FRONT - DISTANCE;
823 3              END;
                ELSE
824 2              DO; FIRST = FRONT;
826 3              IF DISTANCE >= MAX - BACK THEN
827 3                  LASTC = MAXM;
                ELSE
828 3                  LASTC = BACK + DISTANCE;
829 3              END;
830 2      END SETCLIMITS;
/* ----- */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/* read another line of input */
831 1      READLINE: PROCEDURE;
832 2          DECLARE B BYTE;
                /* READ ANOTHER LINE OF INPUT */
833 2          CTRAN: PROCEDURE(B) BYTE;
834 3              DECLARE B BYTE;
                /* CONDITIONALLY TRANSLATE TO UPPER CASE ON INPUT */
835 3              IF UPPER THEN RETURN UTRAN(B);
837 3              RETURN B;
838 3              END CTRAN;
839 2          DO FOREVER;
840 3              IF FRONT >= BACK THEN GO TO OVERFLOW;
842 3              IF (B := CTRAN(GETSOURCE)) = ENDFILE THEN
843 3                  DO; CALL ZERODIST; RETURN;
846 4                  END;
847 3              MEMORY(FRONT) = B;
848 3              CALL INCFRONT;
849 3              IF B = LF THEN
850 3                  DO; CALL INCBASE;
852 4                  RETURN;
853 4                  END;
854 3              END;
855 2          END READLINE;
/* ----- */

```

```

/* write one line out */
856 1      WRITELINE: PROCEDURE;
857 2          DECLARE B BYTE;
858 2          DO FOREVER;
859 3              IF BACK >= MAXM THEN /* EMPTY */
860 3                  DO; CALL ZERODIST; RETURN;

```

```

363 4      END;
364 3      CALL INCBACK;
365 3      CALL PUTDEST(B:=MEMORY(BACK));
366 3      IF B = LF THEN
367 3          DO; CALL INCBASE;
369 4          RETURN;
370 4      END;
371 3      END;
372 2      END WRITELINE;
/* ----- */

```

/* write lines until at least half the
the buffer is empty */

```

373 1      WRHALF: PROCEDURE;
374 2          CALL SETFF;
375 2          DO WHILE DISTNZERO;
376 3              IF HMAX >= (MAXM - BACK) THEN
377 3                  CALL ZERODIST;
378 3              ELSE
379 3                  CALL WRITELINE;
380 2          END;
380 2      END WRHALF;
/* ----- */

```

/* write lines determined by distance
called from W and E commands */

```

331 1      WRITEOUT: PROCEDURE;
332 2          DIRECTION = BACKWARD; FIRST = 1; LASTC = BACK;
333 2          CALL MOVER;
334 2          IF DISTZERO THEN CALL WRHALF;
335 2          /* DISTANCE = 0 IF CALL WRHALF */
336 2          DO WHILE DISTNZERO;
337 3              CALL WRITELINE;
338 3              END;
339 3          IF BACK < LASTC THEN
340 3              DO; DIRECTION = FORWARD; CALL MOVER;
341 3              END;
342 2          END WRITEOUT;
/* ----- */

```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

/* clear memory buffer */

```

377 1      CLEARMEM: PROCEDURE;
378 2          CALL SETFF;
379 2          CALL WRITEOUT;
380 2          END CLEARMEM;
/* ----- */

```

/* clear buffers, terminate edit */

```

901 1      TERMINATE: PROCEDURE;
902 2          CALL CLEARMEM;
903 2          if not newfile then
904 2              DO WHILE (CHAR := GETSOURCE) <> ENDFILE;
905 2          CALL PUTDEST(CHAR);

```

906 3
907 2
908 2

END;
CALL FINIS;
END TERMINATE;

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

/* *****

*** COMMAND PRIMITIVES ***

***** */

/* insert char into memory buffer */

```

707 1  INSERT: PROCEDURE;
710 2      IF FRONT = BACK THEN GO TO OVERFLOW;
712 2      MEMORY(FRONT) = CHAR; CALL INCFRONT;
714 2      IF CHAR = LF THEN CALL INCBASE;
716 2      END INSERT;
/* ----- */

```

/* read a character and check for endfile
or CR */

```

717 1  SCANNING: PROCEDURE BYTE;
718 2      RETURN NOT ((CHAR := READC) = ENDFILE OR
                      (CHAR = CR AND NOT INSERTING));
719 2  END SCANNING;
/* ----- */

```

/* read command buffer and insert characters
into scratch 'til next endfile or CR for
find, next, juxt, or substitute commands
fill at WBE and increment WBE so it
addresses the next empty position of scratch */

```

720 1  COLLECT: PROCEDURE;

721 2      SETSCR: PROCEDURE;
722 3          SCRATCH(WBE) = CHAR;
723 3          IF (WBE := WBE + 1) >= SCRSIZE THEN GO TO OVERFLOW;
725 3          END SETSCR;

726 2      DO WHILE SCANNING;
727 3          IF CHAR = CTLL THEN
728 3              DO; CHAR = CR; CALL SETSCR;
731 4              CHAR = LF;
732 4              END;
733 3          IF CHAR = 0 THEN GO TO BADCOM;
735 3          CALL SETSCR;
736 3          END;
737 2      END COLLECT;
/* ----- */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

/* find the string in scratch starting at

```

938 1  FIND: PROCEDURE(PA,PB) BYTE;
939 2      DECLARE (PA,PB) BYTE;
940 2      DECLARE J ADDRESS,
          (K, MATCH) BYTE;
941 2      J = BACK ;
942 2      MATCH = FALSE;
943 2      DO WHILE NOT MATCH AND (MAXM > J);
944 3          LASTC,J = J + 1; /* START SCAN AT J */
945 3          K = PA ; /* ATTEMPT STRING MATCH AT K */
946 3          DO WHILE SCRATCH(K) = MEMORY(LASTC) AND
              NOT (MATCH := K = PB);
              /* MATCHED ONE MORE CHARACTER */
947 4          K = K + 1; LASTC = LASTC + 1;
948 4          END;
949 3          END;
950 3          IF MATCH THEN /* MOVE STORAGE */
951 2              DO; LASTC = LASTC - 1; CALL MOVER;
952 2              END;
953 2          RETURN MATCH;
954 2          END FIND;
/* ----- */

```

```

          /* set up the search string for F, N, and
          S commands */
958 1  SETFIND: PROCEDURE;
959 2      WBE = 0; CALL COLLECT; WBP = WBE;
960 2      END SETFIND;
/* ----- */

```

```

          /* check for found string in F and S commands */
963 1  CHKFOUND: PROCEDURE;
964 2      IF NOT FIND(0,WBP) THEN /* NO MATCH */ GO TO OVERCOUNT;
965 2      END CHKFOUND;
/* ----- */

```

```

          /* parse read / xfer lib FCB */
967 1  parse$lib: procedure(fcbadr) byte;
968 2      dcl fcbadr address;
969 2      dcl afcb based fcbadr (33) byte;
970 2      dcl b byte;
971 2      b = parse$fcf(fcbadr);
972 2      if afcb(9) = ' ' then
973 2          call move(3,.libfcb(9),fcbadr+9);
974 2      if afcb(1) = ' ' then
975 2          call move(3,.libfcb(1),fcbadr+1);
976 2      return b;
977 2      end parse$lib;
/* ----- */

```

```

          /* print relative position */
978 1  PRINTREL: PROCEDURE;
979 2      CALL PRINTLINE(BASELINE+RELLINE);

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER # _____

```

980 2      END PRINTREL;
/* ----- */

```

```

/* type lines command */
901 1      TYPELINES: PROCEDURE;
982 2      DCL I ADDR;
983 2      DCL C BYTE;
984 2      CALL SETLIMITS;
/* DISABLE THE * PROMPT */
985 2      INSERTING = TRUE;
986 2      IF DIRECTION = FORWARD THEN
987 2          DO; RELLINE = 0; I = FRONT;
990 3          END;
      ELSE
991 2          I = FIRST;
992 2          IF (C := MEMORY(I-1)) = LF then do;
994 3              if COLUMN <> 0 THEN
995 3                  CALL CRLF;
996 3              end;
      else
997 2          relline = relline + 1;

998 2      DO I = FIRST TO LASTC;
999 3      IF C = LF THEN
1000 3          DO;
1001 4          CALL PRINTREL;
1002 4          RELLINE = RELLINE + 1;
1003 4          IF BREAK$KEY THEN GO TO OVERCOUNT;
1005 4          END;
1006 3      CALL PRINTC(C:=MEMORY(I));
1007 3      END;
1008 2      END TYPELINES;
/* ----- */

```

CP/M-86 v.1.1 (vol. II)
 COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # C81-162

```

/* set distance to lines per page (LPP) */
1007 1      SETLPP: PROCEDURE;
1010 2      DISTANCE = LPP;
1011 2      END SETLPP;
/* ----- */

```

```

/* save distance in TDIST */
1012 1      SAVEDIST: PROCEDURE;
1013 2      TDIST = DISTANCE;
1014 2      END SAVEDIST;
/* ----- */

```

```

/* Restore distance from TDIST */
1015 1      RESTDIST: PROCEDURE;
1016 2      DISTANCE = TDIST;
1017 2      END RESTDIST;
/* ----- */

```

/* page command (move n pages and print) */

```

1018 1  PAGE: PROCEDURE;
1019 2      DECLARE I BYTE;
1020 2      CALL SAVEDIST;
1021 2      CALL SETLPP;
1022 2      CALL NOVELINES;
1023 2      I = DIRECTION;
1024 2      DIRECTION = FORWARD;
1025 2      CALL SETLPP;
1026 2      CALL TYPELINES;
1027 2      DIRECTION = I;
1028 2      IF LASTC = MAXM OR FIRST = 1 THEN
1029 2          CALL ZERODIST;
        ELSE
1030 2          CALL RESTDIST;
1031 2      END PAGE;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

/* ----- */

/* wait command (1/2 second time-out) */

```

1032 1  WAIT: PROCEDURE;
1033 2      DECLARE I BYTE;
1034 2      DO I = 0 TO 19;
1035 3          IF BREAK$KEY THEN GO TO RESET;
1037 3          CALL TIME(250);
1038 3          END;
1039 2      END WAIT;

```

/* ----- */

/* set direction to forward */

```

1040 1  SETFORWARD: PROCEDURE;
1041 2      DIRECTION = FORWARD;
1042 2      DISTANCE = 1;
1043 2      END SETFORWARD;

```

/* ----- */

/* append 'til buffer is at least half full */

```

1044 1  APPHALF: PROCEDURE;
1045 2      CALL SETFF; /* DISTANCE = 0FFFFH */
1046 2      DO WHILE DISTNZERO;
1047 3          IF FRONT >= HMAX THEN
1048 3              CALL ZERODIST;
        ELSE
1049 3              CALL READLINE;
1050 3          END;
1051 2      END APPHALF;

```

/* ----- */

/* insert CR LF characters */

```

1052 1  INSCRLF: PROCEDURE;
        /* INSERT CR LF CHARACTERS */
1053 2      CHAR = CR; CALL INSERT;
1055 2      CHAR = LF; CALL INSERT;
1057 2      END INSCRLF;

```


/* ----- */

/* test if invalid delete or
backspace at beginning of inserting */

```
1058 1  ins$errorchk; procedure;
1059 2      if (tcolumn = 255) or (front = 1) then
1060 2          go to reset;
1061 2  end ins$errorchk;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

/* ****

*** COMMAND PARSING ***

*****/

/* test for upper or lower case command
set translate flag (used to determine
if following characters should be translated
to upper case */

```
1062 1  TESTCASE: PROCEDURE;
1063 2      DECLARE T BYTE;
1064 2      TRANSLATE = TRUE;
1065 2      T = LOWERCASE(CHAR);
1066 2      CHAR = UTRAN(CHAR);
1067 2      TRANSLATE = UPPER OR NOT T;
1068 2      END TESTCASE;
```

/* ----- */

/* set translate to false and read next
character */

```
1069 1  READCTAN: PROCEDURE;
1070 2      TRANSLATE = FALSE;
1071 2      CHAR = READC;
1072 2      CALL TESTCASE;
1073 2      END READCTAN;
```

/* ----- */

/* return true if command is only character
not in macro or combination on a line */

```
1074 1  SINGLECOM: PROCEDURE(C) BYTE;
1075 2      DECLARE C BYTE;
1076 2      RETURN CHAR = C AND COMLEN = 1 AND MP = 0;
1077 2      END SINGLECOM;
```

/* ----- */

/* return true if command is only character
not in macro or combination on a line, and
the operator has responded with a 'Y' to a
Y/N request */

```
1078 1  SINGLERCOM: PROCEDURE(C) BYTE;
1079 2      DECLARE (C,i) BYTE;
1080 2      IF SINGLECOM(C) THEN
1081 2          DO forever;
1082 3          CALL CRLF; CALL PRINTCHAR(C);
1084 3          CALL MON1(9, (('-(Y/N)',WHAT,'$'));
1085 3          i = UCASE(READCHAR); CALL CRLF;
```

COPYRIGHT 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER =

```

1037 3      IF i = 'N' THEN GO TO START;
1089 3      IF i = 'Y' THEN
1090 3          RETURN TRUE;
1091 3      END;
1092 2      RETURN FALSE;
1093 2      END SINGLERCOM;
/* ----- */

```

```

/* return true if char is a digit */
1094 1  DIGIT: PROCEDURE BYTE;
1095 2      RETURN (I := CHAR - '0') <= 9;
1096 2      END DIGIT;
/* ----- */

```

```

/* return with distance = number char =
next command */
1097 1  NUMBER: PROCEDURE;
1098 2      DISTANCE = 0;
1099 2      DO WHILE DIGIT;
1100 3          DISTANCE = SHL(DISTANCE,3) +
              SHL(DISTANCE,1) + I;
1101 3      CALL READCTAN;
1102 3      END;
1103 2      END NUMBER;
/* ----- */

```

```

/* set distance to distance relative to
the current line */
1104 1  RELDISTANCE: PROCEDURE;
1105 2      IF DISTANCE > BASELINE THEN
1106 2          DO; DIRECTION = FORWARD;
1108 3          DISTANCE = DISTANCE - BASELINE;
1109 3          END;
      ELSE
1110 2          DO; DIRECTION = BACKWARD;
1112 3          DISTANCE = BASELINE - DISTANCE;
1113 3          END;
1114 2      END RELDISTANCE;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
$ eject
```

```
/* ****
```

```
*** MAIN PROGRAM ***
```

```
***** */
```

```
1115 1  plm:          /* entry of MP/M-86 Interface */

/* INITIALIZE THE SYSTEM */

EDCOMHAND: /* PAST LXI SP,STACK */

ver = version;
/* if low(ver) < cpm3 OR high(ver) < mpmproduct then do;
   call print (,('Requires MP/M 2.0','$'));
   call moni(0,0);
   end; */

/* ***** SET UP MEMORY BUFFER ***** */

/* I/O BUFFER REGION IS 1/8 AVAILABLE MEMORY */
1116 1  NBUF = SHR(MAX := MAXB - .MEMORY,SECTSHF+3) - 1;
/* NBUF IS NUMBER OF BUFFERS - 1 */
1117 1  BUFLNGTH = SHL(DOUBLE(NBUF+1),SECTSHF+1);
/* NOW SET MAX AS REMAINDER OF FREE MEMORY */
1118 1  IF BUFLNGTH + 1024 > MAX THEN
1119 1      DO; CALL PRINT(,('NO MEMORY$'));
1121 2      CALL BOOT;
1122 2      END;

/* REMOVE BUFFER SPACE AND 00 AT END OF MEMORY VECTOR */
1123 1  MAX = MAX - BUFLNGTH - 1;
/* RESET BUFFER LENGTH FOR 1 AND 0 */
1124 1  BUFLNGTH = SHR(BUFLNGTH,1);
1125 1  SBUFAADR = MAXB - BUFLNGTH;
1126 1  DBUFAADR = SBUFAADR - BUFLNGTH;
1127 1  MEMORY(MAX) = 0; /* STOPS MATCH AT END OF BUFFER */
1128 1  MAXM = MAX - 1;
1129 1  HMAX = SHR(MAXM,1);

/* ***** SET UP SOURCE & DESTINATION FILES ***** */

1130 1  parse$fn.buff$adr = .tbuff(1);
1131 1  parse$fn.fcb$adr = .fcb;
1132 1  if (len0 < 0) and (low(ver) = cpm3) then do;
1134 2      call parse; /* password in fcb16 */
1135 2      call move(8,.fcb16,.password);
1136 2      end;
```

```
1137 1  if fcb(1)=' ' then
1138 1      call abort(,('Filename Required$'));
1139 1  if not parse$fcb(.SFCB) then /* parse source fcb */
1140 1      call reboot;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

1141 1      if low(ver)=cpm3 then do;
1143 2          call copydest;          /* copy source fcb to dest */
1144 2          call read$xfcb(.dfcb);
1145 2          protection = dfcb(ex);    /* password protection mode */
1146 2          end;
1147 1      call setdest;                /* parse destination file */
1148 1      command$tail = false;        /* parse fcb from ED command */

```

/* SOURCE AND DESTINATION DISKS SET */

/* IF SOURCE AND DESTINATION DISKS DIFFER, CHECK FOR
AN EXISTING SOURCE FILE ON THE DESTINATION DISK - THERE
COULD BE A FATAL ERROR CONDITION WHICH COULD DESTROY A
FILE IF THE USER HAPPENED TO BE ADDRESSING THE WRONG
DISK */

```

1149 1      IF (SDISK <> DDISK) or not onefile THEN
1150 1          DO;
1151 2          CALL SETDMA(.BUFF);
1152 2          CALL SEARCH(.dfcb);
1153 2          IF DCNT <> 255 THEN /* SOURCE FILE PRESENT ON DEST DISK */
1154 2              CALL ABORT(.( 'Output File Exists, Erase It$' ));
1155 2          END;

```

```

1156 1      RESTART:
          CALL SETUP;
          MEMORY(0) = LF;
          FRONT = 1; BACK = MAXM;
          COLUMN = 0;
          GO TO START;

1162 1      OVERCOUNT: FLAG = POUND; GO TO RESET;

1164 1      BADCOM: FLAG = WHAT; GO TO RESET;

1166 1      OVERFLOW: /* ARRIVE HERE ON OVERFLOW CONDITION (I,F,S COMMAND) */
          FLAG = '>'; go to reset;

1168 1      disk$full$err:
          flag = 'F';
1169 1      err$msg = .diskfull;
1170 1      go to reset;

1171 1      dir$full$err:
          flag = 'F';
1172 1      err$msg = .dirfull;

1173 1      RESET: /* ARRIVE HERE ON ERROR CONDITION */
          PRINTSUPPRESS = FALSE;
          CALL PRINT(.( 'BREAK '$' ));
          CALL PRINTC(FLAG);
          CALL PRINTN(.( ' AT '$' ));
          CALL PRINTC(CHAR);
          if err$msg <> 0 then do;
1174 1          call abort($err);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

1181 2      call print(errmsg);
1182 2      errmsg = 0;
1183 2      end;
1184 1      CALL CRLF;

```

```

1185 1      START:
1186 1      READBUFF = TRUE;
1187 1      MP = 0;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

```

1187 1      DO FOREVER; /* OR UNTIL THE POWER IS TURNED OFF */

/* *****
SIMPLE COMMANDS (CANNOT BE PRECEDED BY DIRECTION/DISTANCE):
    E      END THE EDIT NORMALLY
    H      MOVE TO HEAD OF EDITED FILE
    I      INSERT CHARACTERS
    O      RETURN TO THE ORIGINAL FILE
    R      READ FROM LIBRARY FILE
    Q      QUIT EDIT WITHOUT CHANGES TO ORIGINAL FILE
***** */

1188 2      INSERTING = FALSE;
1189 2      CALL READCTRAN;
1190 2      FLAG = 'E';
1191 2      MI = CDP; /* SAVE STARTING ADDRESS FOR <CR> COMMAND */
1192 2      IF SINGLECOM('E') THEN
1193 2          DO; CALL TERMINATE;
              /* >>>>>>> This code forces a select of the destination
              drive when the edit ends <<<<<<<<<
              I think the user should have a choice in the matter
              so I have commented the code out - Doug 10/14/81
              IF SDISK <> DDISK THEN do;
                  if ddisk <> 0 then
                      ddisk = ddisk - 1;
                  BDISK = (SDISK AND OFOH) OR (DDISK AND OFH); */
1195 3      CALL REBOOT;
1196 3      END;

1197 2      ELSE IF SINGLECOM('H') THEN /* GO TO TOP */
1198 2          DO; CALL TERMINATE;
1200 3          call close(.sfcb);
1201 3          if onefile then do;
              /* PING - PONG DISKS */
              CHAR = DDISK;
1203 4              DDISK = SDISK;
1204 4              SDISK = CHAR;
1205 4              end;
1206 4          else do;
1207 3              call settype(.dfcb,.dtype);
1208 4              call move (16,.dfcb,.sfcb); /* source = destination */
1209 4              onefile = true;
1210 4              end;
1211 4          GO TO RESTART;
1212 3      END;
1213 3

1214 2      ELSE IF CHAR = 'I' THEN /* INSERT CHARACTERS */
1215 2          DO;
1216 3          IF (INSERTING := (CDP = COMLEN) AND (MP = 0)) THEN do;
1218 4              tcolumn = 255; /* tested in insterrorchk routine */
1219 4              distance = 0;
1220 4              direction = backward;
1221 4              if memory(front-1) = LF then

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

1222 4      call printbase;
      else
1223 4      call typelines;
1224 4      end;
1225 3      DO WHILE SCANNING;
1226 4      DO WHILE CHAR <> 0;
1227 5      IF CHAR=CTLU OR CHAR=CTLX OR CHAR=CTLR THEN
      /* LINE DELETE OR RETYPE */
1228 5      DO;
      /* ELIMINATE OR REPEAT THE LINE */
1229 6      IF CHAR = CTLR THEN
1230 6      DO; CALL CRLF;
1232 7      CALL TYPELINES;
1233 7      END;
      ELSE
      /* LINE DELETE */
1234 6      DO; CALL SETLIMITS; CALL SETPTRS;
1237 7      IF CHAR = CTLU THEN
1238 7      DO; CALL CRLF; CALL PRINTNBASE;
1241 8      END;
      ELSE
      /* MUST BE CTLX */
1242 7      DO WHILE COLUMN > SCOLUMN;
1243 8      CALL BACKSPACE;
1244 8      END;
1245 7      END;
1246 6      END;
1247 5      ELSE IF CHAR = CTLH THEN
1248 5      DO;
1249 6      call insterror$chk;
1250 6      IF (TCOLUMN := COLUMN) > 0 THEN
1251 6      CALL PRINTNHAC(' '); /* RESTORE AFT BACKSP */
1252 6      call decfront;
1253 6      if tcolum > scolum then
1254 6      DO; /* CHARACTER CAN BE ELIMINATED */
1255 7      PRINTSUPPRESS = TRUE;
      /* BACKSPACE CHARACTER ACCEPTED */
1256 7      COLUMN = 0;
1257 7      CALL TYPELINES;
1258 7      PRINTSUPPRESS = FALSE;
      /* COLUMN POSITION NOW RESET */
1259 7      IF (OCOLUMN := COLUMN) < SCOLUMN THEN
1260 7      OCOLUMN = SCOLUMN;
1261 7      COLUMN = TCOLUMN; /* ORIGINAL VALUE */
1262 7      DO WHILE COLUMN > OCOLUMN;
1263 8      CALL BACKSPACE;
1264 8      END;
1265 7      END;
      else
1266 6      do;
1267 7      if memory(front-1) = CR then
1268 7      call decfront;
1269 7      call crlf;
1270 7      call typelines;
1271 7      end;
1272 6      CHAR = 0;
1273 6      END;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. B. X 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

1274 5      ELSE IF CHAR = RUBOUT THEN
1275 5          DO; call ins$errorchk;
1277 6          CALL DECFRONT; CALL PRINTC(CHAR:=MEMORY(FRONT));
1279 6          CHAR = 0;
1280 6          END;
1281 5      else if char = LF and memory(front-1) <> CR then
1282 5          do;
1283 6          call printc(CR);
1284 6          call inscrlf;
1285 6          end;
      ELSE
      /* NOT A SPECIAL CASE */
1286 5      DO;
1287 6      IF NOT GRAPHIC(CHAR) THEN
1288 6          DO;
1289 7          CALL PRINTNMAC('^');
1290 7          CALL PRINTNMAC(CHAR + '2');
1291 7          end;
      /* COLUMN COUNT GOES UP IF GRAPHIC */
      /* COMPUTE OUTPUT COLUMN POSITION */
1292 6      if char = CILL and not inserting then
1293 6          call inscrlf;
1294 6      else do;
1295 7          IF MP = 0 THEN
1296 7              DO;
1297 8              IF CHAR >= ' ' THEN
1298 8                  COLUMN = COLUMN + 1;
1299 8              ELSE IF CHAR = TAB THEN
1300 8                  COLUMN = COLUMN + (8 - (COLUMN AND 111B));
1301 8              END;
1302 7              CALL INSERT;
1303 7              END;
1304 6          end;
1305 5      IF CHAR = LF THEN CALL PRINTNMBASE;
1307 5      IF CHAR = CR THEN
1308 5          CALL PRINTNMAC(CHAR:=LF);
      ELSE
1309 5          CHAR = 0;
1310 5          tcolumn = 0;
1311 5          END; /* of while char <> 0 */
1312 4      END; /* of while scanning */
1313 3      IF CHAR <> ENDFILE THEN do; /* MUST HAVE STOPPED ON CR */
1315 4          CALL INSCRLF;
1316 4          column = 0;
1317 4          end;
1318 3      IF INSERTING AND LINESET THEN CALL CRLF;
1320 3      END;

1321 2      ELSE IF SINGLERCOM('0') THEN /* FORGET THIS EDIT */
1322 2          do;
1323 3          call close(.sfcb);
1324 3          GO TO RESTART;
1325 3          end;

1326 2      ELSE IF CHAR = 'R' THEN

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

1327 2      DO; DECLARE I BYTE;
          /* READ FROM LIB FILE */
1327 3      CALL SETRDMA;
1330 3      IF (FLAG := parse$lib(.rfcb)) THEN
1331 3          reading = false;
1332 3      if not reading then do;
1334 4          if not flag then
          /* READ FROM XFER FILE */
1335 4          CALL MOVE(12,.XFCB,.RFCB);
1336 4          RFCB(12), RFCB(32) = 0; /* zero extent, next record */
1337 4          rbp = sectsize;
1338 4          CALL open(.RFCB);
1337 4          reading = true;
1340 4          end;

1341 3      DO WHILE (CHAR := READFILE) <> ENDFILE;
1342 4          CALL INSERT;
1343 4          END;
1344 3      reading = false;
1345 3      call close (.rfcb);
1346 3      END;

```

```

1347 2      ELSE IF SINGLERCOM('Q') THEN
1348 2          DO;
1349 3          CALL DELETE$file(.DFCB);
1350 3          if newfile or not onefile then do;
1352 4              call settype(.dfcb,.dtype);
1353 4              call delete$file(.dfcb);
1354 4              end;
1355 3          CALL REBOOT;
1356 3          END;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

ELSE
  /* MAY BE A COMMAND WHICH HAS AN OPTIONAL DIRECTION AND DISTANCE */
1357 2      DO; /* SCAN A SIGNED INTEGER VALUE (IF ANY) */
1358 3      DCL I BYTE;

1359 3      CALL SETFORWARD;

1360 3      IF CHAR = '-' THEN
1361 3          DO; CALL READCTAN; DIRECTION = BACKWARD;
1364 4          END;

1365 3      IF CHAR = POUND THEN
1366 3          DO; CALL SETFF; CALL READCTAN;
1369 4          END;

1370 3      ELSE IF DIGIT THEN
1371 3          DO; CALL NUMBER;
          /* MAY BE ABSOLUTE LINE REFERENCE */
1373 4          IF CHAR = ':' THEN
1374 4              DO; CHAR = 'L';
1376 5              CALL RELDISTANCE;
1377 5              END;
1378 4          END;

```

```

1379 3      ELSE IF CHAR = ':' THEN /* LEADING COLON */
1380 3          DO; CALL READCTAN; /* CLEAR THE COLON */
1382 4          CALL NUMBER;
1383 4          CALL RELDISTANCE;
1384 4          IF DIRECTION = FORWARD THEN
1385 4              DISTANCE = DISTANCE + 1;
1386 4          END;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

\$ eject

```

1388 3      IF DISTZERO THEN
          DIRECTION = BACKWARD;
/* DIRECTION AND DISTANCE ARE NOW SET */

```

```

/* *****
MAY BE A COMMAND WHICH HAS DIRECTION AND DISTANCE SPECIFIED:
B      BEGINNING/BOTTOM OF BUFFER
C      MOVE CHARACTER POSITIONS
D      DELETE CHARACTERS
K      KILL LINES
L      MOVE LINE POSITION
P      PAGE UP OR DOWN (LPP LINES AND PRINT)
T      TYPE LINES
U      UPPER CASE TRANSLATE
V      VERIFY LINE NUMBERS
<CR>   MOVE UP OR DOWN LINES AND PRINT LINE
***** */

```

```

1389 3      IF CHAR = 'B' THEN
1390 3          DO; DIRECTION = 1 - DIRECTION;
1392 4          FIRST = 1; LASTC = MAXN; CALL MOVER;
1395 4          END;

```

```

1396 3      ELSE IF CHAR = 'C' THEN
1397 3          DO; CALL SETCLIMITS; CALL MOVER;
1400 4          END;

```

```

1401 3      ELSE IF CHAR = 'D' THEN
1402 3          DO; CALL SETCLIMITS;
1404 4          CALL SETPTRS; /* SETS BACK/FRONT */
1405 4          END;

```

```

1406 3      ELSE IF CHAR = 'K' THEN
1407 3          DO; CALL SETLIMITS;
1409 4          CALL SETPTRS;
1410 4          END;

```

```

1411 3      ELSE IF CHAR = 'L' THEN
1412 3          CALL MOVELINES;

```

```

1413 3      ELSE IF CHAR = 'P' THEN /* PAGE MODE PRINT */
1414 3          DO;
1415 4          IF DISTZERO THEN
1416 4              DO; DIRECTION = FORWARD;
1418 5              CALL SETLPP; CALL TYPELINES;
1420 5              END;
          ELSE
1421 4              DO WHILE DISTNZERO; CALL PAGE;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

1423 5          CALL WAIT;
1424 5          END;
1425 4          END;

1426 3          ELSE IF CHAR = 'T' THEN
1427 3              CALL TYPELINES;

1428 3          ELSE IF CHAR = 'U' THEN
1429 3              UPPER = DIRECTION = FORWARD;

1430 3          ELSE IF CHAR = 'V' THEN
1431 3              DO; /* OV DISPLAYS BUFFER STATE */
1432 4                  IF DISTZERO THEN
1433 4                      DO; CALL PRINTVALUE(BACK-FRONT);
1435 5                      CALL PRINTC('/');
1436 5                      CALL PRINTVALUE(MAXM);
1437 5                      CALL CRLF;
1438 5                      END;
1439 4                  ELSE IF (LINESET := DIRECTION = FORWARD) THEN
1440 4                      scolumn = 8;
1441 4                  ELSE
1442 4                      scolumn = 0;
1443 3              END;

1443 3          ELSE IF CHAR = CR THEN /* MAY BE MOVE/TYPE COMMAND */
1444 3              DO;
1445 4                  IF MI = 1 AND MP = 0 THEN /* FIRST COMMAND */
1446 4                      DO; CALL MOVELINES; CALL SETFORWARD; CALL TYPELINES;
1450 5                      END;
1451 4                  END;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

\$ eject

```

1452 3      ELSE IF DIRECTION = FORWARD OR DISTZERO THEN
1453 3      DO;

```

```

/* *****
COMMANDS WHICH ALLOW ONLY A PRECEDING NUMBER:

```

```

A      APPEND LINES
F      FIND NTH OCCURRENCE
M      APPLY MACRO
N      SAME AS F WITH AUTOSCAN THROUGH FILE
S      PERFORM N SUBSTITUTIONS
W      WRITE LINES TO OUTPUT FILE
X      TRANSFER (XFER) LINES TO TEMP FILE
Z      SLEEP

```

```

***** */

```

```

1454 4      IF CHAR = 'A' THEN
1455 4      DO; DIRECTION = FORWARD;
1457 5      FIRST = FRONT; LASTC = MAXM; CALL MOVER;
/* ALL STORAGE FORWARD */
1460 5      IF DISTZERO THEN CALL APPHALF;
/* DISTANCE = 0 IF APPHALF CALLED */
1462 5      DO WHILE DISTNZERO;
1463 6      CALL READLINE;
1464 6      END;
1465 5      DIRECTION = BACKWARD; CALL MOVER;
/* POINTERS REPOSITIONED */
1467 5      END;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

1468 4      ELSE IF CHAR = 'F' THEN
1469 4      DO; CALL SETFIND; /* SEARCH STRING SCANNED
AND SETUP BETWEEN 0 AND WBP-1 IN SCRATCH */
1471 5      DO WHILE DISTNZERO; CALL CHKFOUND;
1473 6      END;
1474 5      END;

```

```

1475 4      ELSE IF CHAR = 'J' THEN /* JUXTAPOSITION OPERATION */
1476 4      DO; DECLARE T ADDRESS;
1478 5      CALL SETFIND; CALL COLLECT;
1480 5      WBJ = WBE; CALL COLLECT;
/* SEARCH FOR STRING 0 - WBP-1, INSERT STRING WBP TO WBJ-1,
AND THEN DELETE UP TO STRING WBJ TO WBE-1 */
1482 5      DO WHILE DISTNZERO; CALL CHKFOUND;
1484 6      /* INSERT STRING */ MI = WBP - 1;
1485 6      DO WHILE (MI := MI + 1) < WBJ;
1486 7      CHAR = SCRATCH(MI); CALL INSERT;
1488 7      END;
1489 6      T = FRONT; /* SAVE POSITION FOR DELETE */
1490 6      IF NOT FIND(WBJ,WBE) THEN GO TO OVERCOUNT;
/* STRING FOUND, SO MOVE IT BACK */
1492 6      FIRST = FRONT - (WBE - WBJ);
1493 6      DIRECTION = BACKWARD; CALL MOVER;

```

```

/* NOW REMOVE THE INTERMEDIATE STRING */
1495 6      call setfront(t);
1496 6      END;
1497 5      END;

1498 4      ELSE IF CHAR = 'M' AND MP = 0 THEN /* MACRO DEFINITION */
1499 4          DO; XP = 255;
1501 5          IF DISTANCE = 1 THEN CALL ZERODIST;
1503 5              DO WHILE (MACRO(XP := XP + 1) := READC) < CR;
1504 6                  END;
1505 5          MP = XP; XP = 0; MT = DISTANCE;
1508 5          END;

1509 4      ELSE IF CHAR = 'N' THEN
1510 4          DO; /* SEARCH FOR STRING WITH AUTOSCAN */
1511 5          CALL SETFIND; /* SEARCH STRING SCANNED */
1512 5          DO WHILE DISTNZERO;
1513 6              /* FIND ANOTHER OCCURRENCE OF STRING */
1514 6              DO WHILE NOT FIND(0,WBP); /* NOT IN BUFFER */
1515 7                  IF BREAK$KEY THEN GO TO RESET;
1516 7                  CALL SAVEDIST; CALL CLEARMEM;
1517 7                  /* MEMORY BUFFER WRITTEN */
1518 7                  CALL APPHALF;
1519 7                  DIRECTION = BACKWARD; FIRST = 1; CALL MOVER;
1520 7                  CALL RESTDIST; DIRECTION = FORWARD;
1521 7                  /* MAY BE END OF FILE */
1522 7                  IF BACK >= MAXM THEN GO TO OVERCOUNT;
1523 7                  END;
1524 7          END;
1525 5      END;

1526 5      END;

1527 5      END;

1528 5      END;

1529 4      ELSE IF CHAR = 'S' THEN /* SUBSTITUTE COMMAND */
1530 4          DO; CALL SETFIND;
1531 5          CALL COLLECT;
1532 5          /* FIND STRING FROM 0 TO WBP-1, SUBSTITUTE STRING
          BETWEEN WBP AND WBE-1 IN SCRATCH */
1533 5          DO WHILE DISTNZERO;
1534 6          CALL CHKFOUND;
1535 6          /* FRONT AND BACK NOW POSITIONED AT FOUND
          STRING - REPLACE IT */
1536 6          call setfront(FRONT - (MI := WBP)); /* BACKED UP */
1537 6          DO WHILE MI < WBE;
1538 7              CHAR = SCRATCH(MI);
1539 7              MI = MI + 1; CALL INSERT;
1540 7          END;
1541 6          END;
1542 5      END;

1543 4      ELSE IF CHAR = 'W' THEN
1544 4          CALL WRITEDOUT;

```

COPYR CHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. =

```

1546 4      DO;
1547 5      flag = parse$lib(.rfcb);
1548 5      xbp = 0;
1549 5      IF DISTZERO THEN
1550 5          DO; /* delete the file */
1551 6          xferon = false;
1552 6          CALL DELETE(.rfcb);
1553 6          if dcnt = 255 then do;
1555 7              flag = '0';
1556 7              go to reset;
1557 7          end;
1558 6      END;
      ELSE
1559 5          do; /* transfer lines */
1560 6          declare i address;

1561 6          if xferon and compare$xfer then
1562 6              call append$xfer;
          else
1563 6              DO;
1564 7              XFERON = TRUE;
1565 7              call move(12,.rfcb,.xfer);
1566 7              xfcnext, xfcbr, xfcbe, xfcbr = 0;
1567 7              CALL MAKE$file(.XFCB);
1568 7              IF DCNT = 255 THEN
1569 7                  goto dir$full$err;
1570 7              END;
1571 6          CALL SETLIMITS;
1572 6          DO I = FIRST TO LASTC;
1573 7              CALL PUTXFER(MEMORY(I));
1574 7          END;
1575 6          call close$xfer;
1576 6          END;
1577 5      END;

1578 4      ELSE IF CHAR = 'Z' THEN /* SLEEP */
1579 4          DO;
1580 5          IF DISTZERO THEN
1581 5              DO; IF READCHAR = ENDFILE THEN GO TO RESET;
1584 6              END;
1585 5              DO WHILE DISTNZERO; CALL WAIT;
1587 6              END;
1588 5          END;
1589 4          ELSE IF CHAR <> 0 THEN /* NOT BREAK LEFT OVER FROM STOP */
/* DIRECTION FORWARD, BUT NOT ONE OF THE ABOVE */
1590 4          GO TO BADCOM;

      END;
      ELSE /* DIRECTION NOT FORWARD */
1592 3          GO TO BADCOM;
1593 3      END;
1594 2      END;
1595 1      END;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

 DEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

255	0000H	1	A.	BYTE BASED(S)	257														
109	0004H	2	A.	WORD PARAMETER AUTOMATIC		110	111												
118	0004H	2	A.	WORD PARAMETER AUTOMATIC		119	120												
261	0004H	2	A.	WORD PARAMETER AUTOMATIC		262	263												
223	0004H	2	A.	WORD PARAMETER AUTOMATIC		224	225												
181	0004H	2	A.	WORD PARAMETER AUTOMATIC		182	183												
113	0004H	2	A.	WORD PARAMETER AUTOMATIC		114	116												
191	0000H	1	A.	BYTE BASED(S)	193														
223	0C44H	19	ABORT.	PROCEDURE STACK=002CH		231	558	609	627	667	678	1138	1154						
30			ADDR	LITERALLY	49 190 255	546	717	811	982										
769	0000H	33	AFCB	BYTE BASED(FCBADR) ARRAY(33)		972	974												
547	0000H	33	AFCB	BYTE BASED(FCBADR) ARRAY(33)		551	586	588											
261	0003H	2	AFCB	WORD PARAMETER AUTOMATIC		262	263												
386	0004H	2	AFCB	WORD PARAMETER AUTOMATIC		387	388												
237	0004H	2	AFCB	WORD PARAMETER AUTOMATIC		238	240												
248	0004H	2	AFCB	WORD PARAMETER AUTOMATIC		249	250	252											
242	0004H	2	AFCB	WORD PARAMETER AUTOMATIC		243	244	246											
371	0EF6H	77	APPENDXFER	PROCEDURE STACK=0012H		1562													
1044	18BFH	34	APPHALF.	PROCEDURE STACK=0040H		1461	1518												
329	0004H	1	B.	BYTE PARAMETER AUTOMATIC		330	333												
857	02E7H	1	B.	BYTE	865 866														
191	0000H	1	B.	BYTE BASED(D)	193														
288	02D5H	1	B.	BYTE	291 273														
296	02D6H	1	B.	BYTE															
833	0004H	1	B.	BYTE PARAMETER AUTOMATIC		834	836	837											
970	02EAK	1	B.	BYTE	971 976														
832	02E6H	1	B.	BYTE	842 847 849														
49	0022H	2	BACK	WORD	526 727 754 768	776	781	785	795	819	826	828	840						
					859 865 876 884 891 910	941	1159	1434	1524										
73	097CH	37	BACKSPACE.	PROCEDURE STACK=0014H		1243	1263												
35	014FH	3	BACKUP	BYTE ARRAY(3) INITIAL		300	406												
43			BACKWARD	LITERALLY	719 748 817	882	1111	1220	1363	1388	1465	1493	1519						
45	0142H		BADCOM	LABEL	934 1164 1590 1592														
42	0016H	2	BASELINE	WORD	491 697 759 762	979	1105	1108	1112										
24	0000H	1	BDISK.	BYTE EXTERNAL(11)															
25	0914H	14	BOOT	PROCEDURE STACK=0008H		215	1121												
166	0836H	39	BREAKKEY	PROCEDURE BYTE STACK=0008H		453	506	1003	1035	1514									
24	0000H	128	BUFF	BYTE ARRAY(128) EXTERNAL(13)		499	500	630	640	1151									
41	0000H	2	BUFFADR.	WORD MEMBER(PARSEFN)	1130														
34	0006H	2	BUFFLENGTH	WORD	289 331 695 1117	1118	1123	1124	1125	1126									
1074	0004H	1	C.	BYTE PARAMETER AUTOMATIC		1075	1076												
96	0004H	1	C.	BYTE PARAMETER AUTOMATIC		97	98	101	103										
455	0004H	1	C.	BYTE PARAMETER AUTOMATIC		456	457	458	460	461									
90	0004H	1	C.	BYTE PARAMETER AUTOMATIC		91	92	94											
430	02DCH	1	C.	BYTE	435 436 437	439	441	450	453										
423	0004H	1	C.	BYTE PARAMETER AUTOMATIC		424	425	426	427										
779	02E5H	1	C.	BYTE	785 787														
983	02EBH	1	C.	BYTE	992 999 1006														
1070	0004H	1	C.	BYTE PARAMETER AUTOMATIC															

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93955

SER =

417	0004H	1	C.	BYTE PARAMETER AUTOMATIC	420	421				
189	0008H	1	C.	BYTE PARAMETER AUTOMATIC	190	192				
254	0004H	1	C.	BYTE PARAMETER AUTOMATIC	255	256				
336	0004H	1	C.	BYTE PARAMETER AUTOMATIC	337	352				
40	01DDH	1	CBP.	BYTE	532	536	537	538	1191	1216
59	0004H	1	CHAR.	BYTE PARAMETER AUTOMATIC	60	63				
65	0004H	1	CHAR.	BYTE PARAMETER AUTOMATIC	66	67	69	71		
81	0004H	1	CHAR.	BYTE PARAMETER AUTOMATIC	82	83	84	85	87	
49	02CFH	1	C.	BYTE	542	553	551	557	559	566
					912	914	918	922	927	929
					1095	1177	1203	1205	1214	1226
					1281	1287	1290	1292	1297	1299
					1365	1373	1375	1379	1389	1396
					1454	1468	1475	1486	1498	1509
					1529	1537	1543	1545	1578	1589
413	0004H	1	C.	BYTE PARAMETER AUTOMATIC	414	417				
963	1A39H	24	CHKFOUND.	PROCEDURE STACK=001CH	1472	1483	1534			
34			CK.	LITERALLY	34					
897	18D1H	11	CLEARMEN.	PROCEDURE STACK=0032H	902	1517				
134	0AA9H	19	CLOSE.	PROCEDURE STACK=000AH	230	322	347	360	394	1200
355	0EA7H	37	CLOSEXFER.	PROCEDURE STACK=0022H	1575					
16	0000H	1	CMDRV.	BYTE EXTERNAL(3)						
920	1956H	47	COLLECT.	PROCEDURE STACK=0038H	960	1479	1481	1532		
38	0155H	1	COLUMN.	BYTE	68	70	74	79	83	524
					1259	1261	1262	1298	1300	1316
40	015CH	123	CONBUFF.	BYTE ARRAY(128)	537	538				
40	015BH	1	CONLEN.	BYTE	536	1076	1216			
47	02CDH	1	COMMANDTAIL.	BYTE INITIAL	541	1148				
362	0ECCH	42	COMPAREXFER.	PROCEDURE BYTE STACK=0002H	1561					
43			CONSIZE.	LITERALLY	44	49				
56	0931H	15	CONIN.	PROCEDURE BYTE STACK=0008H	435					
612	13CDH	19	COPYDEST.	PROCEDURE STACK=000CH	624	1143				
28	0000H	38	COPYRIGHT.	BYTE ARRAY(38) DATA						
2			CPM3.	LITERALLY	234	644	650	1132	1141	
31			CR.	LITERALLY	94	106	437	501	537	555
					1267	1281	1283	1307	1443	1503
105	0A3CH	17	CRLF.	PROCEDURE STACK=0020H	115	226	431	657	658	681
					1086	1184	1231	1239	1269	1319
174	0B5DH	15	CSELECT.	PROCEDURE BYTE STACK=0008H						
30			CTLH.	LITERALLY	76	78	441	1247		
30			CTLL.	LITERALLY	927	1292				
30			CTLR.	LITERALLY	1227	1229				
30			CTLU.	LITERALLY	1227	1237				
30			CTLX.	LITERALLY	439	1227				
833	1821H	25	CTRAM.	PROCEDURE BYTE STACK=0018H	842					
31			CTRLC.	LITERALLY	169					
464	02DDH	1	D.	BYTE	468	471	474			
189	0004H	2	D.	WORD PARAMETER AUTOMATIC	190	191	193	195		
27	0026H	5	DATE.	BYTE ARRAY(5) DATA						
34	0000H	128	DBUFF.	BYTE BASED(DBUFFADR) ARRAY(128)	333					
34	000AH	2	DBUFFADR.	WORD	34	317	333	1126		
30			DCL.	LITERALLY						
39	0157H	1	DCNT.	BYTE	136	140	144	156	276	278
					1153	1553	1568			
34	011AH	1	DDISK.	BYTE AT	1149	1203	1204			
775	16E7H	9	DECBACK.	PROCEDURE STACK=0002H	796					
761	16B4H	9	DECBASE.	PROCEDURE STACK=0002H	773					
770	16CFH	24	DECFRONT.	PROCEDURE STACK=0006H	792	813	1252	1268	1277	

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

555	0055H	14	DEL.	BYTE ARRAY(14) DATA	556	559														
142	0ACFH	19	DELETE	PROCEDURE STACK=000AH	214	240	1552													
237	0C7DH	16	DELETEFILE	PROCEDURE STACK=0014H	244	250	301	1349	1353											
554	1387H	65	DELIM.	PROCEDURE BYTE STACK=0030H	574	576	595													
549	02E1H	1	DELIMITER.	BYTE	556	559	603													
34	011AH	33	DFCB	BYTE ARRAY(33)	34	230	300	301	302	318	322	394	399	401						
					409	410	411	613	617	619	620	625	683	684	685	690	692	694		
					1144	1145	1152	1208	1209	1349	1352	1353								
34	0127H	1	DFUB	BYTE AT																
1094	1CC2H	20	DIGIT.	PROCEDURE BYTE STACK=0002H	1099	1370														
49	02CEH	1	DIRECTION.	BYTE	719	748	780	817	882	893	986	1023	1024	1027	1041	1107				
					1111	1220	1363	1384	1388	1391	1417	1429	1439	1452	1456	1465	1493	1519		
					1523															
52	003CH	15	DIRFULL.	BYTE ARRAY(15) DATA	231	1172														
45	016CH		DIRFULLERR.	'ABEL	1171	1569														
177	0004H	1	DISK	BYTE PARAMETER AUTOMATIC	178	179														
52	004BH	10	DISKFULL	BYTE ARRAY(10) DATA	1169															
45	015AH		DISKFULLERR.	LABEL	323	348	1168													
146	0AE2H	16	DISKREAD	PROCEDURE BYTE STACK=000AH	276	376	636													
150	0AF2H	16	DISKWRITE.	PROCEDURE BYTE STACK=000AH	318	343														
49	001CH	2	DISTANCE	WORD	700	703	706	711	721	736	820	822	826	828	1010	1013				
					1016	1042	1098	1100	1105	1108	1112	1219	1385	1501	1507					
703	158AH	24	DISTNZERO.	PROCEDURE BYTE STACK=0006H	875	888	1046	1421	1462	1471	1482									
					1512	1533	1585													
702	15A0H	15	DIGIZERO	PROCEDURE BYTE STACK=0002H	709	886	1387	1415	1432	1452	1460									
					1549	1580														
			DOUBLE	BUILTIN	1117															
548			DRIVE.	LITERALLY	586															
35	013DH	3	DTYPE.	BYTE ARRAY(3)	410	625	1208	1352												
1	0002H	2322	ED	PROCEDURE STACK=0042H																
1115	0010H		EDCOMMAND.	LABEL																
30			ENDFILE.	LITERALLY	280	291	358	380	391	458	555	567	637	842	904	979				
					918	1313	1341	1582												
48			EOS.	LITERALLY																
295	0D8FH	61	ERASEBAK	PROCEDURE BYTE STACK=0018H	319	344														
609	135CH		ERR.	LABEL	578	585	587	597	604	606										
52	002AH	2	ERRMSG	WORD INITIAL	1169	1172	1178	1181	1182											
37	0010H	2	ERRORCODE.	WORD	649	653	654	660	663											
30			ESC.	LITERALLY	457															
34			EX	LITERALLY	34	643	684	690	694	1145										
453	109EH		EXIT	LABEL	438															
254	0006H	1	F.	BYTE PARAMETER AUTOMATIC	255	257														
31			FALSE.	LITERALLY	34	35	36	50	172	306	367	466	523	562	564					
					610	621	714	739	804	942	1070	1092	1148	1173	1188	1258	1331	1344		
					1551															
142	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	143	144														
138	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	139	140														
134	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	135	136														
130	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	131	132														
122	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	123	124														
17	0000H	1	FCB.	BYTE ARRAY(1) EXTERNAL(4)	34	276	613	619	649	660	666	668								
					670	1131	1137													
198	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	199	201														
204	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	205	206														
185	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	186	187														
158	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	159	160														
154	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	155	156														

CP/M-86 v.1.1 (vol. II)
 COPYRIGHT © 1981. 1
 DIGITAL RESEARCH
 11111 11111 11111 11111 11111 11111 11111 11111 11111 11111
 PACIFIC GROVE, CA 9

SER - C81-162

150	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	151	152												
146	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	147	148												
18	0000H	1	FCB16.	BYTE ARRAY(1) EXTERNAL(5)	1135													
767	0004H	2	FCBADR.	WORD PARAMETER AUTOMATIC	968	969	971	972	973	974	975							
545	002EH	2	FCBAUR.	WORD PARAMETER	546	547	551	569	570	586	588							
41	0002H	2	FCBADR.	WORD MEMBER(PARSEFN)	1131													
229	0C57H	19	FERR.	PROCEDURE STACK=0030H	279	396	687											
254	0C8DH	32	FILL.	PROCEDURE STACK=0008H	433	569	570											
268	0D00H	91	FILLSOURCE.	PROCEDURE STACK=0034H	290													
933	19A5H	129	FIND.	PROCEDURE BYTE STACK=0018H		964	1490	1513										
385	0F43H	119	FINIS.	PROCEDURE STACK=0034H	907													
49	0024H	2	FIRST.	WORD	750	754	791	821	822	825	883	991	998	1028	1392	1457		
					1492	1520	1572											
44	02C6H	1	FLAG.	BYTE	126	1162	1164	1166	1168	1171	1175	1190	1330	1334	1547	1555		
549	02E0H	1	FLAG.	BYTE	552	564	565	608										
31			FOREVER.	LITERALLY	839	858	1081	1187										
43			FORWARD.	LITERALLY	780	893	986	1024	1041	1107	1384	1417	1429	1439	1452			
					1456	1523												
49	0020H	2	FRONT.	WORD	722	751	765	771	772	787	791	795	812	820	822	825		
					840	847	910	912	989	1047	1059	1158	1221	1267	1278	1281	1434	1457
					1489	1492	1535											
34			FS.	LITERALLY	34													
12	0000H	1	FUNC.	BYTE PARAMETER	13													
8	0000H	1	FUNC.	BYTE PARAMETER	9													
4	0000H	1	FUNC.	BYTE PARAMETER	5													
498	1176H	34	GETCMD.	PROCEDURE BYTE STACK=0002H		542	626											
429	101EH	136	GETPASSWD.	PROCEDURE STACK=002AH	656													
287	0D66H	41	GETSOURCE.	PROCEDURE BYTE STACK=0038H		842	904											
540	126AH	27	GETUC.	PROCEDURE STACK=0034H	565	572	580	589	594	599								
90	09E4H	53	GRAPHIC.	PROCEDURE BYTE STACK=0006H		98	1287											
			HIGH.	BUILTIN	654													
32	0004H	2	HMAX.	WORD	876	1047	1129											
82	02D2H	1	I.	BYTE	83	86												
1560	003EH	2	I.	WORD	1572	1573												
33	0040H	1	I.	BYTE	1095	1100												
1328	02F0H	1	I.	BYTE														
1079	02EFH	1	I.	BYTE	1085	1087	1089											
616	02E2H	1	I.	BYTE	618	619												
1033	02EDH	1	I.	BYTE	1034													
932	003AH	2	I.	WORD	939	991	992	998	1006	SER								
1358	02F1H	1	I.	BYTE														
717	0030H	2	I.	WORD	722	727	733	734	740	745	750	755						
1019	02ECH	1	I.	BYTE	1023	1027												
430	02DBH	1	I.	BYTE	434	436	443	446										
549	02DFH	1	I.	BYTE	551	575	577	584	593	596								
363	02DAH	1	I.	BYTE	364	365	366											
356	02D9H	1	I.	BYTE	357													
309	02D7H	1	I.	BYTE	316													
269	02D4H	1	I.	BYTE	274	281												
767	16C6H	9	INCBACK.	PROCEDURE STACK=0002H		782	864											
758	16ABH	9	INCBASE.	PROCEDURE STACK=0002H		786	851	868	915									
764	16BDH	9	INCFRONT.	PROCEDURE STACK=0002H		788	848	913										
12	0000H	2	INFO.	WORD PARAMETER	14													
8	0000H	2	INFO.	WORD PARAMETER	10													
4	0000H	2	INFO.	WORD PARAMETER	6													
1052	1BE1H	21	INSCRLF.	PROCEDURE STACK=000AH		1284	1293	1315										
1058	1BE6H	25	INSEPRORCH.	PROCEDURE STACK=0002H		1249	1274											

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER =

909	1903H	41	INSERT	PROCEDURE STACK=0006H	1054	1056	1302	1342	1487	1539
46	02CAH	1	INSERTING.	BYTE	486	519	918	985	1188	1216 1292 1318
52	002BH	17	INVALID.	BYTE ARRAY(17) DATA	609	627				
82	02D3H	1	J.	BYTE	86					
940	003BH	2	J.	WORD	941	943	944			
940	02E8H	1	K.	BYTE	945	946	947			
464	002CH	2	K.	WORD	465	467	468	469	470	
717	0032H	2	K.	WORD	724	729	733	740		
717	0034H	2	L.	WORD	723	728	733			
			LAST	BUILTIN	556					
49	0026H	2	LASTC.	WORD	751	755	781	819	827	828 884 891 944 946 948 953
					998	1028	1393	1458	1572	
30			LCA.	LITERALLY	421					
30			LCZ.	LITERALLY	421					
20	0000H	1	LENO	BYTE EXTERNAL(7)		1132				
22	0000H	1	LEN1	BYTE EXTERNAL(9)						
31			LF	LITERALLY	69	94	107	533	733	772 785 849 866 914 931
					992	999	1055	1157	1221	1281 1305 1308
35	0140H	12	LIBFCB	BYTE ARRAY(12) INITIAL		214	973	975		
42	01DEH	1	LINESET.	BYTE INITIAL	481	524	1318	1439		
30			LIT.	LITERALLY	30	43	548			
717	02E4H	1	LOOPING.	BYTE	731	732	736	739	744	
			LOW.	BUILTIN	234	313	390	468	644	650 653 663 1132 1141
419	0FE6H	29	LOWERCASE.	PROCEDURE BYTE STACK=0006H		425	1065			
43			LPP.	LITERALLY	1010					
717	0036H	2	M.	WORD	733	734	745			
44	01DFH	128	MACRO.	BYTE ARRAY(128)	517	1503				
43			MACSIZE.	LITERALLY	44					
154	0802H	19	MAKE	PROCEDURE STACK=000AH		252				
248	0CA7H	22	MAKEFILE	PROCEDURE STACK=001AH		685	1567			
940	02E9H	1	MATCH.	BYTE	942	943	946	951	956	
32	0000H	2	MAX.	WORD	826	1116	1118	1123	1127	1128
24	0000H	2	MAXB	WORD EXTERNAL(12)		1116	1125			
40	015AH	1	MAXLEN	BYTE	163	164				
32	0002H	2	MAXM	WORD	526	728	827	859	876	943 1028 1128 1129 1159 1393 1436
					1458	1524				
34			MD	LITERALLY	34	643				
778	16F0H	107	MEMMOVE.	PROCEDURE STACK=000CH		801	804			
	0000H		MEMORY	BYTE ARRAY(0)	733	772	785	787	795	847 865 912 946 992
					1006	1116	1127	1157	1221	1267 1278 1281 1573
44	02C8H	1	MI	BYTE	1191	1445	1484	1485	1486	1535 1536 1537 1538
717	02E3H	1	MIDDLE	BYTE	733	737				
208	0004H	1	MODE	BYTE PARAMETER AUTOMATIC		209	210			
4	0000H		MON1	PROCEDURE EXTERNAL(0) STACK=0000H				26	63	111 120 160 179
					183	187	206	210	221	1084
8	0000H		MON2	PROCEDURE BYTE EXTERNAL(1) STACK=0000H				54	57	124 136 140
					144	148	152	156	167	169 175 201
12	0000H		MON3	PROCEDURE WORD EXTERNAL(2) STACK=0000H				132	218	
189	0B9FH	37	MOVE	PROCEDURE STACK=0008H		200	263	388	613	625 973 975 1135
					1209	1335	1565			
778	0004H	1	MOVEFLAG	BYTE PARAMETER AUTOMATIC		779	783	793		
806	1771H	11	MOVELINES.	PROCEDURE STACK=001AH		1022	1412	1447		
800	175BH	11	MOVER.	PROCEDURE STACK=0010H		808	885	894	954	1394 1399 1459 1466
					1494	1521				
336	0FB9H	24	MOVEUP	PROCEDURE STACK=000EH		405	409			
44	02C7H	1	MP	BYTE	415	494	504	508	1076	1186 1216 1295 1445 1498 1505
				LITERALLY						

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

FARMINGDALE, CA 95955

44	001AH	2	HT	WORD	510	512	1507											
309	02D8H	1	N.	BYTE	313	316												
34	0109H	1	NBUF	BYTE	274	281	1116	1117										
47	02CCH	1	NCMD	BYTE INITIAL		499	500											
34	000EH	2	NDEST.	WORD	311	313	317	325	331	333	334	390	696					
35	013BH	1	NEWFILE.	BYTE INITIAL		298	679	903	1350									
810	0004H	2	NEWFRONT	WORD PARAMETER AUTOMATIC				811	812									
34			NR	LITERALLY		34	643											
34	000CH	2	NSOURCE.	WORD	271	275	280	283	289	291	292	695						
1097	1CD6H	49	NUMBER	PROCEDURE STACK=0038H				1372	1382									
435	1044H		NXTCHR	LABEL	447													
35	013CH	1	ONEFILE.	BYTE INITIAL		297	403	621	623	664	677	1149	1201	1210	1350			
122	0A7DH	28	OPEN	PROCEDURE STACK=000AH				373	1338									
130	0A99H	16	OPENFILE	PROCEDURE WORD STACK=000AH				649	660									
45	0136H		OVERCOUNT.	LABEL	507	513	965	1004	1162	1491	1525							
45	014EH		OVERFLOW	LABEL	841	911	924	1166										
938	0003H	1	PA	BYTE PARAMETER AUTOMATIC				939	945									
1018	1B43H	61	PAGE	PROCEDURE STACK=0034H				1422										
220	0C35H	15	PARSE.	PROCEDURE STACK=0008H				1134										
545	1285H	226	PARSEFCB	PROCEDURE BYTE STACK=0038H				617	971	1139								
41	0012H	4	PARSEFN.	STRUCTURE	221	1130	1131											
967	1A51H	63	PARSELIB	PROCEDURE BYTE STACK=003EH				1330	1547									
19	0000H	2	PASS0.	WORD EXTERNAL(6)														
21	0000H	2	PASS1.	WORD EXTERNAL(8)														
34	010AH	16	PASSWORD	BYTE ARRAY(16) INITIAL				200	235	433	436	446	1135					
938	0004H	1	PB	BYTE PARAMETER AUTOMATIC				939	946									
3	0010H		PLN.	LABEL PUBLIC			1115											
43			POUND.	LITERALLY		1162	1365											
113	0A5DH	16	PRINT.	PROCEDURE STACK=0026H				202	225	432	680	1120	1174	1181				
81	09A1H	67	PRINTABS	PROCEDURE STACK=0016H				100	103									
490	1159H	12	PRINTBASE.	PROCEDURE STACK=002CH				496	528	1222								
96	0A19H	35	PRINTC	PROCEDURE STACK=001CH				106	107	417	474	476	484	485	487			
					488	530	533	1006	1175	1177	1180	1278	1283	1435				
59	0940H	26	PRINTCHAR.	PROCEDURE STACK=000AH				71	1083									
479	1126H	51	PRINTLINE.	PROCEDURE STACK=0028H				491	527	979								
109	0A4DH	16	PRINIM	PROCEDURE STACK=000AH				116	1176									
413	0FD2H	20	PRINTNMAC.	PROCEDURE STACK=0022H				1251	1289	1290	1308							
493	1165H	17	PRINTNMBASE.	PROCEDURE STACK=0030H				1240	1306									
978	1A90H	16	PRINTREL	PROCEDURE STACK=002CH				1001										
36	0152H	1	PRINTSUPPRESS.	BYTE INITIAL		61	1173	1255	1258									
463	10C9H	73	PRINTVALUE	PROCEDURE STACK=0022H				483	1434	1436								
30			PROC	LITERALLY		189	254	429	498	540	545	550	554	716	764	770		
					775	778	800	803	806	810	816							
36	0154H	1	PROTECTION	BYTE INITIAL				688	690	1145								
550	1367H	32	PUTC	PROCEDURE STACK=0002H				579	598									
329	0E34H	36	PUTDEST.	PROCEDURE STACK=0022H				391	865	905								
336	0E5CH	79	PUTXFER.	PROCEDURE STACK=001EH				358	1573									
38	0158H	1	QCOLUMN.	BYTE	1259	1260	1262											
34	0062H	1	RBP.	BYTE	633	638	640	1337										
118	0A6DH	16	READ	PROCEDURE STACK=000AH				164										
46	02C8H	1	READBUFF	BYTE	521	523	536	1185										
503	1198H	210	READC.	PROCEDURE BYTE STACK=0030H				543	918	1071	1503							
53	0722H	15	READCHAR	PROCEDURE BYTE STACK=0008H				520	1085	1582								
162	0B25H	17	READCON.	PROCEDURE STACK=000EH				531										
1069	1C37H	19	READCTAN.	PROCEDURE STACK=0034H				1101	1189	1362	1368	1381						
632	1449H	56	READFILE	PROCEDURE BYTE STACK=0016H				1341										
34	0108H	1	READING.	BYTE INITIAL		1331	1332	1339	1344									

COPYRIGHT © 1981.1
DIGITAL RESEARCH
P.O. BOX 570
PACIFIC GROVE, CA 93950

SER. #

831	17E1H	64	READLINE	PROCEDURE STACK=003CH	1049	1463			
204	0BEDH	16	READXFCB	PROCEDURE STACK=000AH	1144				
212	0C10H	22	REBOOT	PROCEDURE STACK=000EH	227	451	1140	1195	1355
1104	1D07H	40	RELDISTANCE	PROCEDURE STACK=0002H	1376	1383			
42	0013H	2	RELLINE	WORD	719	743	979	988	997 1002
158	0B15H	16	RENAME	PROCEDURE STACK=000AH	246				
242	0C3DH	26	RENAMEFILE	PROCEDURE STACK=001AH	407	411			
45	0177H		RESET	LABEL	127	1036	1060	1163	1165 1167 1170 1173 1515 1556 1583
45	011AH		RESTART	LABEL	1156	1212	1324		
1015	1B38H	11	RESTDIST	PROCEDURE STACK=0002H	1030	1522			
433	102BH		RETRY	LABEL	440	444			
340	0E62H		RETRY	LABEL	345				
317	0DF1H		RETRY	LABEL	320				
208	0BFDH	19	RETURNERRORS . . .	PROCEDURE STACK=000AH	646	652			
34	0041H	33	RFCB	BYTE ARRAY(33) INITIAL	366	636	1330	1335	1336 1338 1345 1547
					1552	1565			
34			RO	LITERALLY	666				
			ROL	BUILTIN	666	668	670		
43			RUBOUT	LITERALLY	1274				
254	0008H	2	S	WORD PARAMETER AUTOMATIC	255	257	258		
189	0006H	2	S	WORD PARAMETER AUTOMATIC	190	191	193	194	
1012	1B2DH	11	SAVEDIST	PROCEDURE STACK=0002H	1020	1516			
34	0000H	128	SBUF	BYTE BASED(SBUFFADR) ARRAY(128)		280	291		
34	0008H	2	SBUFFADR	WORD	34	275	280	291	1125 1126
717	192CH	42	SCANNING	PROCEDURE BYTE STACK=0034H	926	1225			
38	0156H	1	SCOLUMN	BYTE INITIAL	1242	1253	1259	1260	1440 1441
44	025FH	100	SCRATCH	BYTE ARRAY(100)	922	946	1486	1537	
43			SCRSIZE	LITERALLY	44	923			
34	0000H	1	SDISK	BYTE EXTERNAL(4) AT	1149	1204	1205		
138	0ABCH	19	SEARCH	PROCEDURE STACK=000AH	1152				
24			SECTSHF	LITERALLY	313	1116	1117		
24			SECTSIZE	LITERALLY	34	283	325	338	357 379 633 1337
177	0B6CH	19	SELECT	PROCEDURE STACK=000AH					
185	0B8FH	16	SETATTRIBUTE . . .	PROCEDURE STACK=000AH	401				
816	1790H	31	SETCLIMITS	PROCEDURE STACK=0002H	1398	1403			
615	13DBH	98	SETDEST	PROCEDURE STACK=003CH	1147				
101	0B7FH	16	SETDMA	PROCEDURE STACK=000AH	235	266	275	317	630 1151
699	1595H	11	SETFF	PROCEDURE STACK=0002H	874	898	1045	1367	
958	1A26H	19	SETFIND	PROCEDURE STACK=003CH	1470	1478	1511	1531	
1040	1BAFH	16	SETFORWARD	PROCEDURE STACK=0002H	1359	1448			
810	177CH	20	SETFRONT	PROCEDURE STACK=000CH	1495	1535			
716	15D2H	217	SETLIMITS	PROCEDURE STACK=0004H	807	984	1235	1408	1571
1009	1B22H	11	SETLPP	PROCEDURE STACK=0002H	1021	1025	1418		
233	0C6AH	19	SETPASSWORD	PROCEDURE STACK=000EH	239	245	251	400	647 659 691
303	1766H	11	SETPTRS	PROCEDURE STACK=0010H	1236	1404	1409		
629	143DH	12	SETDMA	PROCEDURE STACK=000EH	635	1329			
921	1985H	32	SETSCR	PROCEDURE STACK=0002H	930	935			
261	0CDDH	23	SETTYPE	PROCEDURE STACK=0010H	300	302	406	410	683 1208 1352
642	1431H	276	SETUP	PROCEDURE STACK=0034H	1156				
265	0CF4H	12	SETXDMA	PROCEDURE STACK=000EH	340	375			
34	0000H	33	SFCB	BYTE ARRAY(33) EXTERNAL(4) AT	405	406	407	643	1139 1200 1209
					1323				
			SHL	BUILTIN	1100	1117			
			SHR	BUILTIN	313	1116	1124	1129	
1074	1C4CH	46	SINGLECOM	PROCEDURE BYTE STACK=0006H	1080	1192	1197		
1078	1C7AH	72	SINGLERCOM	PROCEDURE BYTE STACK=0026H	1321	1347			
45	018AH		START	LABEL	1088	1141	1185		

COPYRIGHT © 1981.

DIGITAL RESEARCH

P. O. BOX 570

PACIFIC GROVE, CA 93950

SER. =

34		SY	LITERALLY	399	668															
36	0153H	1 SYS.	BYTE INITIAL	397	672															
1477	003CH	2 T.	WORD	1489	1495															
1063	02EEH	1 T.	BYTE	1065	1067															
30		TAB.	LITERALLY	83	84	94	1180	1299												
23	0000H	1 TBUF.	BYTE ARRAY(1) EXTERNAL(10)					1130												
40	01DCH	1 TCBP	BYTE																	
38	0157H	1 TCOLUMN.	BYTE	1059	1218	1250	1253	1261	1310											
49	001EH	2 TDIST.	WORD	1013	1016															
35	014CH	3 TEMPFL	BYTE ARRAY(3) INITIAL					302	683											
901	18DCH	39 TERMINATE.	PROCEDURE STACK=003CH					1194	1199											
1062	1C0FH	42 TESTCASE	PROCEDURE STACK=0016H					1072												
		TIME	BUILTIN	1037																
50	02D0H	1 TRANSLATE.	BYTE INITIAL	459	1064	1067	1070													
31		TRUE	LITERALLY	35	42	93	170	304	369	473	552	560	672	679						
				712	731	801	839	858	985	1064	1081	1090	1185	1187	1210	1255	1339			
				1564																
65	095AH	34 TTYCHAR.	PROCEDURE STACK=0010H			76	77	78	87											
981	1A0H	130 TYPELINES.	PROCEDURE STACK=0030H			1026	1223	1232	1257	1270	1419	1427	1449							
34		UB	LITERALLY	34																
423	1003H	27 UCASE.	PROCEDURE BYTE STACK=000CH			435	460	542	543	1085										
50	02D1H	1 UPPER.	BYTE INITIAL	835	1067	1429														
34		US	LITERALLY	670																
455	10A6H	35 UTRAN.	PROCEDURE BYTE STACK=0012H			517	520	538	640	836	1066									
479	0004H	2 V.	WORD PARAMETER AUTOMATIC			480	483													
463	0004H	2 V.	WORD PARAMETER AUTOMATIC			464	468	469												
51	002CH	2 VER.	WORD	234	644	650	1115	1132	1141											
217	0C26H	15 VERSION.	PROCEDURE WORD STACK=0008H			1115														
1032	1B80H	47 WAIT	PROCEDURE STACK=000CH			1423	1586													
44	02C4H	1 WBE.	BYTE	922	923	959	961	1480	1490	1492	1536									
44	02C5H	1 WBJ.	BYTE	1480	1485	1490	1492													
44	02C3H	1 WBP.	BYTE	961	964	1484	1513	1535												
31		WHAT	LITERALLY	1084	1164															
873	186BH	38 WRHALF	PROCEDURE STACK=002AH			887														
308	0DCCH	73 WRITEDEST.	PROCEDURE STACK=001CH			332	393													
856	183AH	49 WRITELINE.	PROCEDURE STACK=0026H			878	889													
831	1891H	64 WRITEDOUT	PROCEDURE STACK=002EH			899	1544													
198	0BC4H	41 WRITEXFCB.	PROCEDURE STACK=002CH			692														
34	0105H	1 XBP.	BYTE	338	350	352	353	357	379	380	1548									
34	0086H	128 XBUF.	BYTE ARRAY(128)			266	352	380												
34	0063H	33 XFCB	BYTE ARRAY(33) INITIAL			34	343	347	360	366	373	376	1335							
				1565	1567															
34	0070H	1 XFCBC.	BYTE AT																	
34	006FH	1 XFCBE.	BYTE AT	341	372	1566														
34	0084H	1 XFCBEXT.	BYTE INITIAL	341	372	1566														
34	0071H	1 XFCBM.	BYTE AT																	
34	0083H	1 XFCBR.	BYTE AT	342	374	378	1566													
34	0085H	1 XFCBREC.	BYTE INITIAL	342	374	378	1566													
34	0107H	1 XFERON	BYTE INITIAL			213	1551	1561	1564											
44	02C9H	1 XP	BYTE	508	515	517	1500	1503	1505	1506										
464	02DEH	1 ZERO	BYTE	466	471	473														
705	15AFH	11 ZERODIST	PROCEDURE STACK=0002H			844	861	877	1029	1048	1502									
310	0E27H	11 ZN	PROCEDURE STACK=0002H			315	327													
270	0D5BH	11 ZN	PROCEDURE STACK=0002H			273	285													

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH

P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. =

MODULE INFORMATION:

CODE AREA SIZE = 1D2FH 7471D
CONSTANT AREA SIZE = 011EH 286D
VARIABLE AREA SIZE = 02F2H 754D
MAXIMUM STACK SIZE = 0042H 66D
2653 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. =

